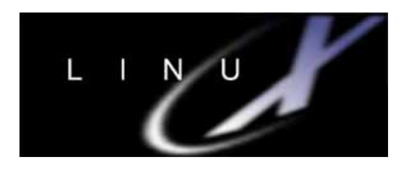
HACK X CRACK: CREA CON NOSOTROS UN GENERADOR DE IPS EN PHP

PC PASO a PASO

INSTALAMOS LINUX



+



EN LA CONSOLA DE BILL GATES

NUMERO 15



1ª PARTE GRATIS EN NUESTRA WEB

Curso de Visual Basic





EN BANDEJA



Nº 15 -- P.V.P. 4,5 EUROS



HACK X CRACK - HACK X CRACK - HACK X CRACK

3 SERVIDORES ON LINE PARA TUS PRACTICAS DE HACK

LOS CUADERNOS DE HACK X CRACK

www.hackxcrack.com

DESTRIPAMOS EL MESSENGER

SESION MSN POR TELNET

COMANDOS MSN - PUERTOS MSN

MSN - CHAT - ¿MSN-HIJACKING?

MSN -FTP - MD5SUM

LISTANDO LAS IP DE TELEFONICA EN PHP

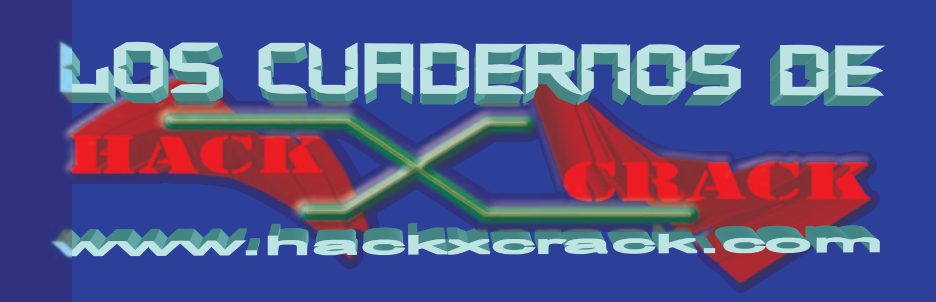
AUTENTIFICACION POR DESAFIO MD5

LOS SERVIDORES DEL MSN: DISPATCH NOTIFICATION Y SWITCHBOARD

LOS MEJORES ARTÍCULOS GRATIS EN NUESTRA WEB

PC PASO A PASO: CONVIERTE LA CONSOLA XBOX DE MICROSOFT EN UN PC





EDITORIAL: EDITOTRANS S.L.
C.I.F: B43675701
PERE MARTELL Nº 20, 2º - 1ª
43001 TARRAGONA (ESPAÑA)

Director Editorial
I. SENTIS
E-mail contacto
director@editotrans.com
Título de la publicación
Los Cuadernos de HACK X CRACK.
Nombre Comercial de la publicación
PC PASO A PASO
Web: www.hackxcrack.com
Dirección: PERE MARTELL Nº 20, 2º - 1ª.
43001 TARRAGONA (ESPAÑA)



Director de la Publicación
J. Sentís

E-mail contacto
director@hackxcrack.com

Diseño gráfico:
J. M. Velasco

E-mail contacto:
grafico@hackxcrack.com

Redactores
AZIMUT, ROTEADO, FASTIC, MORDEA, FAUSTO, ENTROPIC, MEIDOR, HASHIMUIRA, BACKBONE, ZORTEMIUS, AK22, DORKAN, KMORK, MAILA, TITINA, SIMPSIM... ... ...

Contacto redactores
redactores@hackxcrack.com

Colaboradores
Mas de 130 personas: de España, de Brasil, de Argentina, de Francia, de Alemania, de Japón y algún Estadounidense.

E-mail contacto
colaboradores@hackxcrack.com

Imprime
I.G. PRINTONE S.A. Tel 91 808 50 15

DISTRIBUCIÓN:
SGEL, Avda. Valdeparra 29 (Pol. Ind.)
28018 ALCOBENDAS (MADRID)
Tel 91 657 69 00 FAX 91 657 69 28
WEB: www.sgel.es

TELÉFONO DE ATENCIÓN AL CLIENTE: 977 22 45 80
Petición de Números atrasados y Suscripciones (Srta. Genoveva)
HORARIO DE ATENCIÓN: DE 9:30 A 13:30
(LUNES A VIERNES)

NUMERO 15 -- PRINTED IN SPAIN
PERIOCIDAD MENSUAL
Deposito legal: B.26805-2002
Código EAN: 8414090202756

EDITORIAL

UN NUEVO CAMINO

ENERO: Un nuevo año se presenta y un camino desconocido ha sido puesto ante cada uno de nosotros. Es tu elección ponerte a andar o esperar hasta el año que viene.

Los que empiecen a andar, cuando llegue el final mirarán hacia atrás y casi no reconocerán a aquel que empezó a andar. Ese extraño ser que ahora somos nada tendrá que ver con el que seremos, a cada paso creceremos, tras cada suspiro cambiaremos y cada elemento de tiempo consumido darán forma al ser en que nos convertiremos.

Hay quienes se niegan a andar, hay quienes no quieren cambiar; pero se engañan, porque el tiempo moldea incluso a los que no quieren crecer ni mejorar.

Tú eliges, tuya es la elección de quedarte sentado mirando como eres moldeado o participar en ese cambio.

NOSOTROS ya hemos empezado a andar… ¿Qué harás Tú?

P.D. El cambio no puede evitarse, pero con el esfuerzo suficiente SI puedes dirigirlo.

INDICE

IDICE DE ANUNCIANTES

CURSO DE PHP (II)

APRENDE A PROGRAMAR TU PROPIO GENERADOR DE IPS

- Vamos a programar en PHP un generador de IPs
- Descubriremos las IP de Telefónica
- Trataremos los tipos, separadores, cadenas, arrays y estructuras de control de PHP

Continuamos con el curso de PHP, el anterior número sirvió para romper el hielo y comenzar a programar en el fantástico mundo del PHP. A muchos de vosotros os habrá sabido a poco el capítulo anterior, por ello hemos orientado el presente articulo con un ejemplo muy práctico, vamos a crear un generador de IP. Mejor que sigas leyendo.... seguro que te vas a divertir.

Si, has leído bien, vamos a explicar como programar un generador de IP, es decir, un pequeño programita que generará todas las IPs de un rango dado. Este programa nos será muy útil para futuros programas.

Interpretando el PHP

En el foro muchos de vosotros habéis preguntando sobre el tema de los separadores de PHP, así que a continuación se va explicar un poco más a fondo para evitar confusiones ya que en anterior capítulo no quedó muy claro para muchos.

Para interpretar un archivo, PHP simplemente ejecuta en el servidor las instrucciones que encuentra entre los caracteres especiales que delimitan el inicio y final del código. Es decir, para que el interprete PHP ejecute el código es necesario que este se encuentre delimitado entre caracteres especiales, de esta forma el código PHP puede encontrarse embebido en páginas HTML. En el capítulo anterior para simplificar se trabajó con los caracteres especiales <? Para inicio de código PHP y ?> Para fin de código PHP.

Realmente existen 4 conjuntos de etiqueta que sirven para delimitar los bloques de código PHP, de estas 4 solo 2 están siempre disponibles.

Ejemplo 1

```
<?php
print ("Hola mundo");
?>
```

Ejemplo 2

```
<script language="php">
print ("Hola mundo");
</script>
```

Ambos ejemplos son ejecutados por el intérprete dando como resultado el mensaje "Hola mundo", así que para programar puedes utilizar conjunto de etiquetas. El resto de conjuntos pueden configurarse en el archivo PHP.ini para ser aceptados o no por el intérprete.

Otros lenguajes interpretados como ASP (Active Server Pages) utiliza el conjunto de separadores **<%** (para inicio), **código ASP** y **%>** (para fin de código ASP). Cambiando el archivo PHP.ini se puede configurar para que las instrucciones PHP puedan ser embebidas como en ASP. De esta forma el siguiente ejemplo sería interpretado como PHP:

```
<% print ("Hola mundo"); %>
```

Si la versión de PHP instalada es el igual o superior a 3.0.4 entonces ya viene configurado en el archivo PHP.ini y podrás utilizar sin problemas los conjuntos propios del ASP. Tal vez, muchos de vosotros estéis familiarizados con el ASP y os sea más cómodo utilizar estos separadores.

Pero aún se puede abreviar el ejemplo primero con el punto y coma (**;**), por ejemplo:

```
<?php print ("Hola Mundo"); ?>
```

puede pasar a

```
<? print ("Hola Mundo"); ?>
```

Es importante saber que no es recomendable usar el formato abreviado cuando se desarrollenaplicaciones o librerías, con intención de distribuirlas, ya que por defecto no siempre el intérprete acepta este sistema. Por lo general el interprete PHP.ini en Windows acepta el formato abreviado, pero en Linux no siempre será así. De todos modos esto se configura en el archivo PHP.ini.

¿Tu interprete no acepta el formato abreviado?, si al poner el programita <? print ("Hola Mundo"); > no aparece el mensaje Hola Mundo en el navegador, y además al visualizar la página aparece el código PHP, entonces el interprete no está configurado para aceptar el formato abreviado. **¿Qué hacemos entonces para activar el modo abreviado?**.

Hay que editar el archivo de configuración PHP.INI y buscar el parámetro short_open_tag. Para activar el formato abreviado hay que colocar short_open_tag on.

Tipos

PHP soporta los siguientes tipos:

- Matrices
- Números en punto flotante.
- Enteros
- Cadenas

El tipo de una variable normalmente no lo indica el programador; en su lugar, lo decide PHP en tiempo de ejecución dependiendo del contexto en el que se utilice esa variable.

Enteros

Estos son los llamados **números enteros**. Cuando veas un término que no conoces, lo más sencillo (y esto forma parte de tu aprendizaje) es ir a www.google.com y ver de qué estamos hablando. Un número entero es del tipo -59, -3, 0, 1, 5, 78, 34567, etc., es decir, los números naturales (números positivos sin decimales como por ejemplo 1,2,3,4,5... ...), sus opuestos (números negativos sin decimales, como por ejemplo -1, -2, -3, -4, -5...) y el cero. Si esto te viene de nuevo, seguro que hiciste campana el día que explicaban los "conjuntos de números" en clase de "mates" ;p

Los enteros se pueden especificar usando una de las siguientes sintaxis:

```
$a = 1234; # número en base decimal
$a = -123; # un número negativo
$a = 0123; # número en base octal (equivalente al 83 decimal)
$a = 0x12; # número hexadecimal (equivalente al 18 decimal)
```



Ya tocamos...

Ya tocamos en la revista los números en base binaria, no es cuestión ahora de explicar la base octal y la base hexadecimal porque en www.google.com existen infinidad de Webs donde puedes aprender a utilizarlos (y operar con ellos). Podríamos escribir 10 páginas explicando cómo sumar, restar, multiplicar y dividir en octal y hexadecimal, pero estaríamos desperdiciando 10 páginas, en google tienes todo lo que necesitas si quieres adentrarte en ese tema. De momento, para este artículo no lo necesitamos :)

Números en punto flotante

Los números en punto flotante ("double") se pueden especificar utilizando cualquiera de las siguientes sintaxis:

```
$a = 1.234;
$a = 1.2e3;
```

Cadenas

Las cadenas van delimitadas entre comillas dobles. $cadena="Hola Mundo".

Si la cadena está encerrada entre dobles comillas ("), las variables que estén dentro de la cadena serán expandidas (sujetas a ciertas limitaciones de interpretación).

```
<?
$cadena1="Tu revista favorita es:";
$cadena2=" hackxcrack.com";
print $cadena1.$cadena2 ."<br>"; // imprime en
      pantalla Tu revista favorita es: hackxcrack.com
print "$cadena1$cadena2<br>"; // imprime en
      pantalla Tu revista favorita es: hackxcrack.com
?>
```

Como se puede ver el resultado es el mismo, una tercera posibilidad podría ser el siguiente programa:

```
<?
$cadena1="Tu revista favorita es:";
$cadena1="$cadena1 hackxcrack.com";
print $cadena1 ."<br>"; // imprime en pantalla
                Tu revista favorita es: hackxcrack.com
?>
```

Es decir, la variable contenida en una cadena es sustituida por el valor anteriormente asignado.

Como en C y en Perl, el carácter de barra invertida ("\") se puede usar para especificar caracteres especiales:

\n	Nueva línea
\r	Retorno de carro
\t	Tabulación horizontal
\\	Barra invertida
\$	Signo del dólar
\"	Comillas dobles
\[0-7]{1,3}	la secuencia de caracteres que coincida con la expresión regular es un carácter en notación octal
\x[0-9A-Fa-f]{1,2}	la secuencia de caracteres que coincida con la expresión regular es un carácter en notación hexadecimal

La segunda forma de delimitar una cadena de caracteres usa el carácter de comilla simple ("'"). Cuando una cadena va encerrada entre comillas simples, los únicos caracteres de escape que serán comprendidos son "\\" y "\'". Esto es por convenio, así que se pueden tener comillas simples y barras invertidas en una cadena entre comillas simples. Las variables no se expandirán dentro de una cadena entre comillas simples. Si tienes dudas pregunta en el foro de Hack x Crack (www.hackxcrack.com) y pondremos unos ejemplos.

Arrays

Un array es un conjunto de "cosas" del mismo tipo. Por ejemplo una lista de IPs, una lista de los nombres de nuestros amigos o la lista de la compra.

Se puede crear una array usando las funciones list() o array(), o se puede asignar el valor de cada elemento del array de manera explícita. El siguiente ejemplo es un array de IPs:

```
<?php
$ip[0]="127.0.0.1";
$ip[1]="127.0.0.2";
$ip[2]="127.0.0.3";
print $ip[2]; // imprime 127.0.0.3
?>
```

También se puede crear un array simplemente añadiendo valores al array. Cuando se asigna un valor a una variable array usando corchetes vacíos, el valor se añadirá al final del array.

```
<?php
$ip[]="127.0.0.1";
$ip[]="127.0.0.2";
$ip[]="127.0.0.3";
print $ip[2]; // imprime 127.0.0.3
?>
```

Con count() se puede saber el número de elementos que contiene un array, por ejemplo:

```
<?php
$ip[]="127.0.0.1";
$ip[]="127.0.0.2";
$ip[]="127.0.0.3";
print count($ip); // imprime 3
?>
```

El siguiente programa muestra el contenido de un array, la sentencia for será explicada posteriormente pero analizando el programa ya podéis haceros una idea para que sirve.

```
<?php
$ip[]="127.0.0.1";
$ip[]="127.0.0.2";
$ip[]="127.0.0.3";
$cantidad=count($ip); // $cantidad toma el valor 3
for ($cuenta=0;$cuenta<$cantidad;$cuenta=$cuenta+1) {
print $ip[$cuenta]."<br>"; // Da como salida el elemento
                              con indice $cuenta
}
?>
```

Por ahora es suficiente, más adelante veremos otras funciones para poder manipular los arrays. A medida que vayamos creando ejemplos iremos explicando nuevas funciones de las variables array.

Estructuras de control

Las instrucciones, también llamadas sentencias, en un lenguaje de programación son la forma que tiene el programador de llevar a cabo determinadas acciones en una aplicación. Las instrucciones PHP (y en la gran mayoría de los lenguajes) se dividen en instrucciones de condición e instrucciones de bucle.

Las instrucciones de condición nos permiten especificar que partes del código serán ejecutadas si se cumple una determinada acción.

IF

La construcción IF es una de las sentencias más importantes a la hora de programar decisiones, con IF podemos ejecutar parte de código en función de uno o varios criterios.

PHP caracteriza la sentencia IF de manera similar a C.

```
<?php
$revista="hackxcrack";
if ($revista=="hackxcrack") print " PC Paso a Paso";
?>
```

Se considera expresión a $revista=="hackxcrack", es la parte del IF en dónde se evalúa la condición, da como resultado un valor booleano True (verdadero) o False (falso), es decir, en el anterior ejemplo evalúa la expresión preguntándose de si la variable revista es igual a la cadena hackxcrack, en caso de que de que la comparación sea verdadera (True), entonces el resultado obtenido es "PC Paso a Paso".

Obviamente si la comparación resulta falsa (False) no obtendremos ningún resultado.

Hay que hacer notar lo siguiente, para asignar con un valor a una variable se utiliza el =, pero en una condición se utiliza el doble =, fíjate que en la condición hay dos iguales ==.

Esto puede ser un error común.

Así que, para asignaciones de variables hay que poner el = y para condiciones hay que poner ==.

```
<?php
$revista=”hackxcrack”;
if ($revista==”hackxcrack”) print “ PC Paso a Paso”;
?>
```

También se puede utilizar otras expresiones como:

```
<?php
$edad=”20”;
if ($edad>=18) print “Eres mayor de edad”;
if ($edad<18) print “Eres menor de edad”;
?>
```

Si $edad es mayor o igual a 18 entonces muestra el mensaje **Eres mayor de edad**, si $edad es menor de 18 muestra **Eres menor de edad**.

Las condiciones, como es lógico, pueden ser combinadas utilizando los operadores lógicos.

Recordamos que los operadores lógicos son and (&&), or (||) y xor.

```
<?php
if ((5>4) && (9<6)) || 5=5) print (“Esto es un ejemplo”);
?>
```



¿Operadores Lógicos?...

- *¿Operadores Lógicos? ¿Qué es “eso”?*
- Busca en google, busca en google … …

No es cuestión de dar ahora una clase magistral de -resultados booleanos- A medida que pongamos ejemplos lo verás tu mismo, así que, para los que no tienen ni idea, haremos un par de ejemplos sencillísimos.

Ejemplo 1: and (&&)

El siguiente programa en PHP imprimirá “HAS ACERTADO” en caso de que la condición se cumpla. Lo vemos con un ejemplo y quedará aclarado.

```
<?php
if ((50>5) && (900>3) && (5=5)) print (“HAS ACERTADO”);
?>
```

El programa dice que, en caso de que 50 sea mayor que 5 y 900 sea mayor que 3 y 5 sea igual a 5, imprime en la pantalla HAS ACERTADO. En este caso todo se cumple y por lo tanto en tu monitor verás el mensajito :)

Venga, no seas perezoso, en el número anterior ya te enseñamos a ejecutar el código PHP (y si no lo compraste puedes pedir la revista anterior en www.hackxcrack.com o descargar ese artículo de la misma Web, en la sección “artículos liberados”.

ANÉCDOTA ¿el error del novato?:

En la redacción de PC PASO A PASO recibimos 4 mails donde los lectores nos decían que cuando intentaban ejecutar el código les salía un error, en la pantalla del navegador veían mensajes como este: “Parse error: parse error, unexpected T_STRING in c:\appserv\www\e1.php on line 2”

Los colaboradores de Hack x Crack intercambiamos más de 50 mails con los 4 lectores intentando averiguar dónde estaba el problema… imposible… todo era correcto, el código de la revista estaba bien y nos funcionaba en todos los equipos pero a los lectores no… era desesperante. Finalmente, uno de los colaboradores pidió a los lectores que explicasen paso a paso cómo escribían el código y cómo lo ejecutaban con el navegador.

Pues bien, encontramos el error!!! ¿Cuántas veces hemos dicho en Hack x Crack que WORD no es precisamente un buen sitio donde escribir código de programación?!!! Nuestros lectores escribían el código en Word y después hacían un copiar/pegar en el Block de Notas de Windows!!!

Cuando utilizas WORD, aunque parezca que lo que escribes es lo que aparece en pantalla, es mentira, en realidad el propio WORD introduce automáticamente un montón de

"marcas" que no ves PERO que sí afectarán de forma impredecible cuando lo copies el texto (el código) a otro programa (en este caso el Block de Notas). Por favor, utiliza directamente el Block de Notas o cualquier otro editor de Texto Plano para escribir el código!!! ;p

Ejemplo 2: and (&&)

```
<?php
if ((50>5) && (900>950) && (5=5)) print ("HAS ACERTADO");
?>
```

En este caso no verás nada en la pantalla al ejecutarlo porque falla una de las condiciones, 900 NO ES mayor que 950, por lo tanto la condición no se cumple. Como puedes ver, TODO debe ser cierto para que el operador && de el visto bueno y obtengamos el mensajito :)

Ejemplo 3: or (&&)

```
<?php
if ((50>5) || (900>950) || (5=5)) print ("HAS ACERTADO");
?>
```

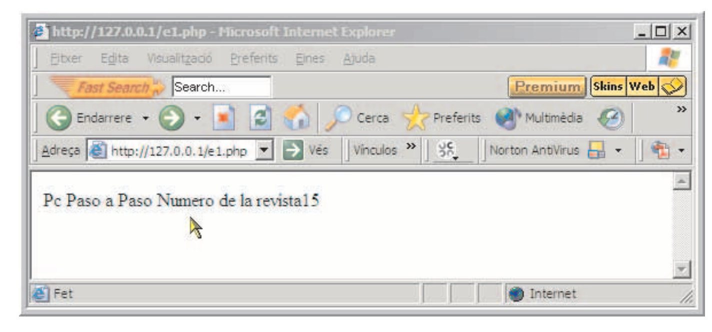
El programa dice que, en caso de que 50 sea mayor que 5 o 900 sea mayor que 950 o 5 sea igual a 5, imprime en la pantalla HAS ACERTADO. En este caso tenemos que 900 no es mayor que 950, pero no importa, el operador || dará por válida la condición siempre que al menos uno de los operadores sea cierto. Como 50 es mayor que 5 ya está, veremos el mensajito "HAS ACERTADO" en la pantalla

Podríamos seguir así hasta cansarnos y construir las tablas de la verdad de cada operador, pero mejor volvemos al artículo y lo vemos todo a medida que hagamos prácticas :)

Hasta ahora en los ejemplos con IF se ha colocado una sentencia para ejecutar en caso de resultado True, si se desea ejecutar más sentencias entonces se pueden agrupar utilizando las **llaves { .. sentencias ... }**

```
<?php
$revista="hackxcrack";
if ($revista=="hackxcrack") {
        print " PC Paso a Paso";
        $numero=15;
        print " Numero de la revista".$numero;
}
?>
```

Venga, ejecuta el código y verás el resultado :)



Hasta ahora hemos visto como ejecutar sentencias si se cumple una condición, pero ¿y si queremos ejecutar un grupo de sentencias si no se cumple la condición?, el siguiente ejemplo muestra como realizarlo:

```
<?php
if ($revista=="hackxcrack") {
        print ("Es tu revista favorita.");
} else {
        print ("Mejor que compres la revista HackxCrack.");
}
?>
```

En el ejemplo anterior la sentencia else se ejecuta solamente si la expresión if se evalúa como **FALSE**, es decir, son excluyentes.

En lenguaje coloquial sería: "Si (if) revista es igual a (==) hackxcrack saca por pantalla (print) Es tu revista favorita, en caso contrario (else) saca por pantalla Mejor que compres la revista HackxCrack.

Se pueden establecer tantas condiciones excluyentes como queramos, ¿cómo? pues muy sencillo, con la sentencia elseif podemos poner todas las condiciones excluyentes que nos

vengan a la imaginación (con cierta lógica, claro, ahora lo veremos).

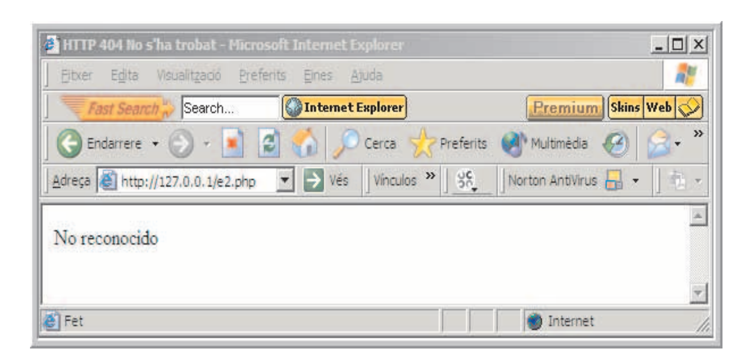
En algunas ocasiones es necesario realizar varias comparaciones seguidas en IF diferentes, por ejemplo:

```
<?php
if ($ip=="127.0.0.1") print ("loopback");
if ($ip=="212.163.200.113") print ("Knet");
if ($ip=="80.58.4.44") print ("Telefonica");
if ($ip=="62.117.144.107") print ("MedTelecom");
if ($ip=="62.151.49.95") print ("Ya");
if ($ip=="66.98.60.123") print ("Codetel");
?>
```

El intérprete irá ejecutando todos los if y esto supone una carga de tiempo para el intérprete, es decir, colocar muchos if seguidos hace que el programa se ralentice, imagina tener cientos de if seguidos, el intérprete tendrá que evaluarlos todos aunque el if con resultado True sea el primero. Para evitarlo, PHP dispone de la sentencia } elseif {

```
<?php
if ($ip=="127.0.0.1") {
        print ("loopback");
} elseif  ($ip=="212.163.200.113") {
        print  ("Knet");
} elseif  ($ip=="80.58.4.44") {
        print ("Telefonica");
} elseif  ($ip=="62.117.144.107") {
        print ("MedTelecom");
} elseif  ($ip=="62.151.49.95") {
        print ("Ya");
} elseif  ($ip=="66.98.60.123") {
        print ("Codetel");
} else {
        print ("No reconocido");
}
?>
```

El intérprete dejará de comprobar las condiciones (if) en cuanto encuentre una condición verdadera (TRUE). En este caso, como se desconoce el valor de la variable **$ip**, llegaríamos al final y en pantalla saldría el mensaje No reconocido.



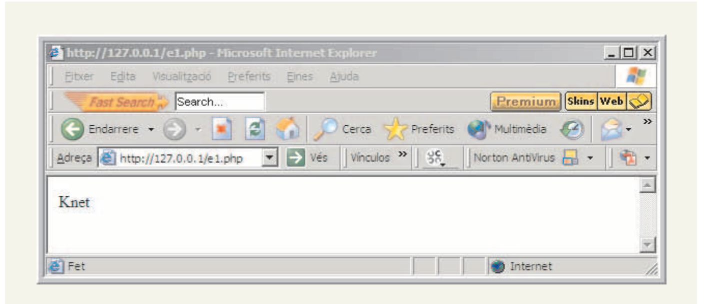


Vamos a...

Vamos a comprobar que realmente el intérprete de PHP se detiene cuando encuentra una condición verdadera (TRUE). Le daremos a la variable **$ip** el valor 127.0.0.1 y pondremos dos condiciones TRUE (en las dos se cumplirá la condición). Fíjate en el siguiente código.

```
<?php
$ip="127.0.0.1";
if ($ip=="66.66.66.66") {
print ("loopback");
} elseif  ($ip=="127.0.0.1") {
print  ("Knet");
} elseif  ($ip=="80.58.4.44") {
print ("Telefonica");
} elseif  ($ip=="127.0.0.1") {
print ("MedTelecom");
} elseif  ($ip=="62.151.49.95") {
print ("Ya");
} elseif  ($ip=="66.98.60.123") {
print ("Codetel");
} else {
print ("No reconocido");
}
?>
```

Ejecútalo y obtendremos la siguiente pantallita



Como puedes ver en el código (en rojo), hay dos condiciones verdaderas (TRUE) pero el interprete solo ha reaccionado ante la primera escribiendo "Kent".



En PHP...

En PHP, también se puede escribir 'else if' (con dos palabras) y el comportamiento sería idéntico al de un 'elseif' (una sola palabra). El significado sintáctico es ligeramente distinto (si estas familiarizado con C, es el mismo comportamiento), pero la línea básica es que ambos resultarían tener exactamente el mismo comportamiento.

Sintaxis alternativa de estructura de control

PHP ofrece una sintaxis alternativa para algunas de sus estructuras de control. La forma básica de la alternativa es cambiar abrir-llave **{** por dos puntos **:** y cerrar llave **}** por **endif**.

```
<?php if ($a==5): ?>
A es igual a 5
<?php endif; ?>
```

La sintaxis alternativa se aplica a else y también a elseif. La siguiente es una estructura if con elseif y else en el formato alternativo:

```
<?php
if ($ip=="127.0.0.1") :
        print ("loopback");
elseif ($ip=="212.163.200.113"):
        print ("Knet");
elseif ($ip=="80.58.4.44"):
        print ("Telefonica");
elseif ($ip=="62.117.144.107"):
        print ("MedTelecom");
elseif ($ip=="62.151.49.95"):
        print ("Ya");
elseif ($ip=="66.98.60.123"):
        print ("Codetel");
else:
        print ("No reconocido");
endif
?>
```

Instrucción SWITCH

En determinadas ocasiones se hace necesario ejecutar diversas condiciones con el fin de comparar un dato entre varias posibilidades.

Como se ha visto en el apartado anterior, esto se puede realizar mediante el uso de la instrucción elseif.

Esta tarea se puede realizar de manera mucho más sencilla, utilizando la instrucción **switch...case**.

```
<?php
switch ($ip) {
case "127.0.0.1": // Comprueba si es la IP de loopback
        print ("loopback");
        break;
case "212.163.200.113"): // Comprueba si es la IP del ISP Knet
        print ("Knet");
        break;
case "80.58.4.44"): // Comprueba si es la IP del ISP Telefonica
        print ("Telefonica");
        break;
case "62.117.144.107"): // Comprueba si es la IP del ISP MedTelecom
        print ("MedTelecom");
        break;
case "62.151.49.95"):
        print ("Ya");
        break;
```

```
case "66.98.60.123":
        print ("Codetel");
        break;
default: // si no se encuentra el IP muestra el siguiente mensaje
        print ("No reconocido");
        break;
}
?>
```

La instrucción **switch** comprueba el valor de **$ip** y lo compara con cada uno de los valores de las posibilidades de **case**.

Cuando la comparación es cierta las instrucciones contenidas en el bloque del case son ejecutadas hasta encontrar el **break**, en caso de que no se cumpla ningún **case** entonces se ejecuta las sentencias contenidas en el **default**.

Instrucciones de bucle

Las sentencias bucles son de gran utilidad pues permiten ejecutar un bloque de instrucciones un número determinando de veces, en función de una o varias condiciones.

Existen dos formas de crear bucles: los bucles creados a partir de la sentencia **for ()**, que repite un número de veces fijo un bloque de instrucciones y los bucle creados por **While** que a diferencia de for() repite un bucle de instrucciones un función a una o varias condiciones.

La sentencia While

El significado de una sentencia **while** es simple. Le dice a PHP que ejecute la(s) sentencia(s) anidada(s) repetidamente, mientras la expresión **while** se evalúe como verdadera (**TRUE)**.

El valor de la expresión es comprobado cada vez al principio del bucle, así que incluso, si este valor cambia durante la ejecución de la(s) sentencia(s) anidada(s), la ejecución no parará hasta el fin de la iteración (cada vez que PHP ejecuta las sentencias en el bucle es una iteración).

A veces, si la expresión **while** se evalúa como falsa (**FALSE)** desde el principio de todo, la(s) sentencia(s) anidada(s) no se ejecutarán ni siquiera una vez.

```
<?php
 $i = 0;
while ($i <= 255) {
        print "IP: 80.67.23.".$i."<br>";
        $i=$i+1;
}
?>
```

En pantalla aparecerá un listado de 256 IPs, se ejecuta el bloque de sentencias contenidas entre las llaves mientras que la variable $i sea menor o igual a 255.

En lenguaje coloquial sería:

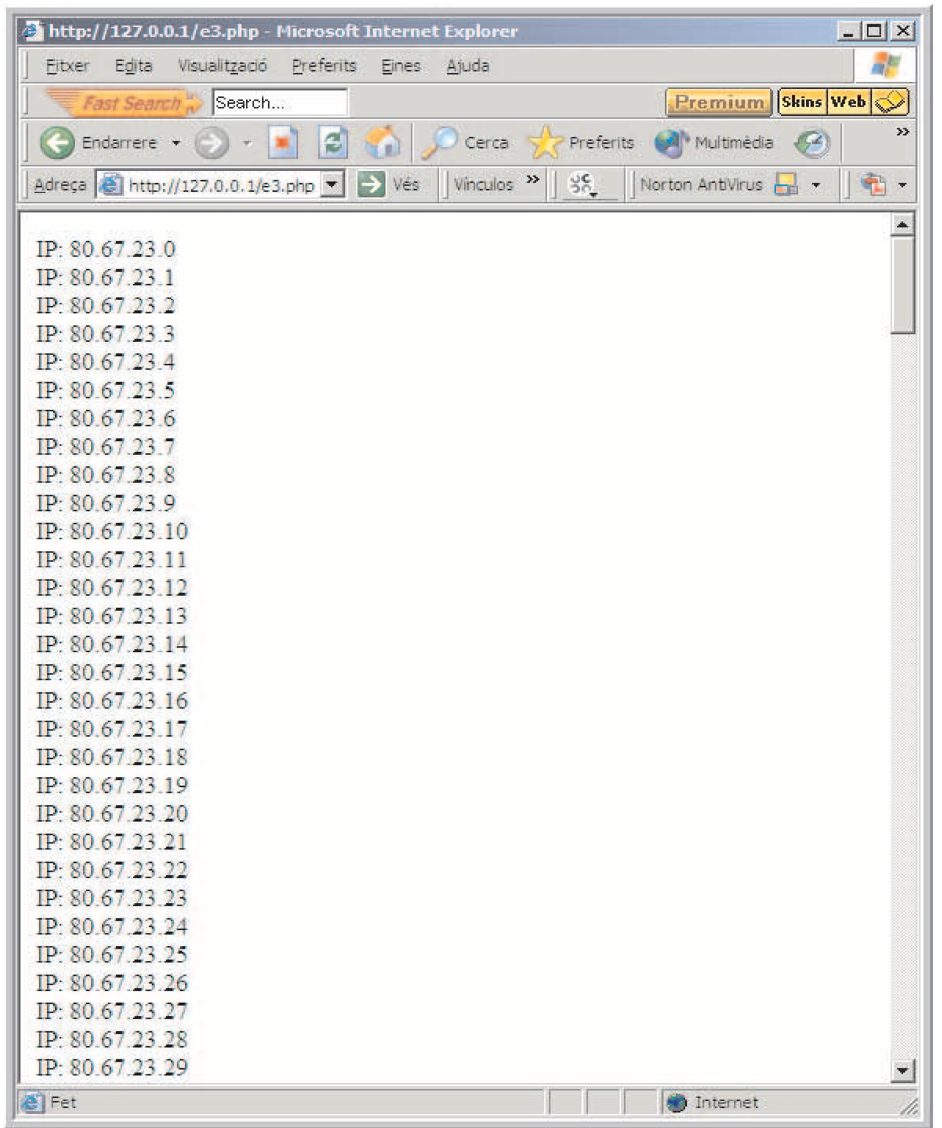
1.- La variable (**$i**) es igual (**=**) a **0**

2.- Mientras (**while**) la variable (**$i**) sea menor o igual (**<=**) que **255**, saca por pantalla (**print**) la IP.

Es decir, comprueba la veracidad de la condición (**$i <= 255**) y si es TRUE continua.

3.- la variable (**$i**) es igual a la variable (**$i**) mas (**+**) uno (**1**). Es decir, le suma una unidad a la variable (**$i**). En este punto el interprete vuelve al punto 2 y comprueba el valor de la variable (**$i**).

Este proceso se repetirá hasta que ($i) valga 256. En ese momento, cuando en el punto 2 se compruebe la condición y resulte falsa (FALSE) se detendrá en programa.



La sentencia Do ... while

Esta sentencia es similar a la anterior comentada While excepto que las condiciones se comprueban al final de la iteración, es decir, justo lo contrario de While que la comparación es al principio.

```
<?php
 $i = 0;
do {
            print "IP: 80.67.23.".$i."<br>";
            $i=$i+1;
 } while ($i<=255);
?>
```

Esto implica que aunque no se cumpla la condición el contenido del bucle siempre se ejecutará al menos una vez.

En lenguaje coloquial:

1.- La variable (**$i**) es igual a (**=**) cero (**0**)

2.- Hacer (**do**) lo que hay a continuación, es decir, imprimir (**print**) la IP y sumar uno a la variable (**$i=$i+1**)

3.- Mientras que (**while**) la variable (**$i**) sea menor o igual (**<=**) que 255. Es decir, que seguirá imprimiendo la IP hasta que la variable valga 253, en ese momento se parará la ejecución del código.

La sentencia For

Los bucles For son los más complejos y completos, son los más usados, ya que permiten ejecutar un número de veces fijo un bloque de instrucciones. La sintaxis de un bucle for es:

for (expr1; expr2; expr3) sentencia

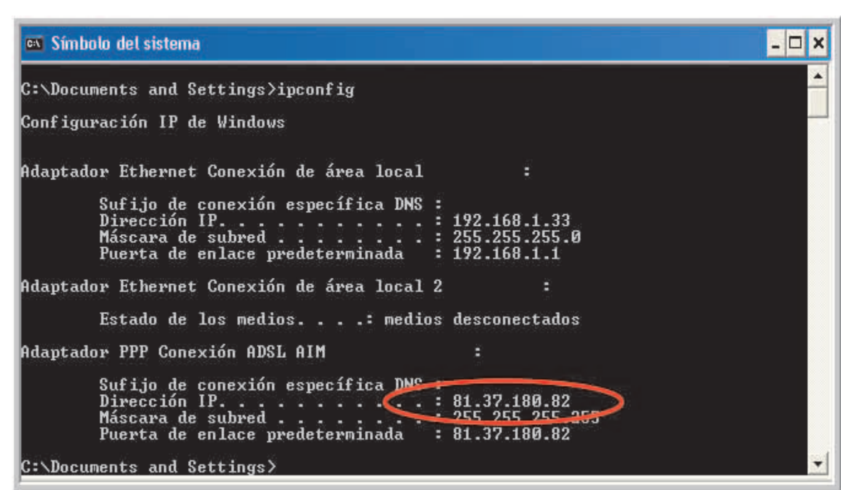
La primera expresión (expr1) se evalúa (ejecuta) incondicionalmente una vez (y solo una vez) al principio del bucle.

Al comienzo de cada iteración, se evalúa expr2 . Si se evalúa como **TRUE**, el bucle continúa y las sentencias anidadas se ejecutan. Si se evalúa como **FALSE**, la ejecución del bucle finaliza.

Al final de cada iteración, se evalúa (ejecuta) expr3.

Cada una de las expresiones puede estar vacía. Que expr2 esté vacía significa que el bucle debería correr indefinidamente (PHP implícitamente lo considera como **TRUE**, al igual que C). Esto puede que no sea tan inútil como se podría pensar, puesto que a menudo se quiere salir de un bucle usando una sentencia break condicional en vez de usar la condición de for.

```
<?php
for ($cuenta=0;$cuenta<=255;
$cuenta=$cuenta+1) {
print (“IP: 80.67.23.$cuenta<br>”);
}
?>
```



Ya lo deberías tener claro, pero por si acaso:

1.- Se ejecuta la expresión 1 ($cuenta=0), es decir, ponemos el valor de la variable $cuenta a cero.

2.- Se ejecuta la expresión 2 ($cuenta<=255). Si es verdadera continuamos, si es falsa el programa se para.

3.- En caso de que fuese verdadera se ejecuta la expresión 3 ($cuenta=$cuenta+1), es decir, sumamos una unidad a la variable $cuenta.

4.- Imprime (print) la IP y volvemos al punto 2, es decir, volvemos a ejecutar la expresión 2.

El proceso continuará hasta que la variable ($cuenta) valga 253.

Ejemplo práctico

Tanta teoría aburre un poco, ¿verdad?, vamos a poner en práctica las sentencias aprendidas, para ello vamos a desarrollar un programa en PHP que muestre en pantalla todas las IP de nuestro proveedor de conexión (ISP). En próximos capítulos aprovecharemos este programa para realizar un escaneo de IPs en busca de servidores.

Lo primero es conocer la IP que nos ha asignado el proveedor. La tarea es muy sencilla y ha sido explicada en números anteriores, pero en caso de duda lo volveremos a repetir.

Abre la ventana de comandos de MS-DOS y pon el comando **ipconfig**.

Fíjate en la IP que aparece en el adaptador PPP, en este caso, es una conexión ADSL y la IP es 81.37.180.82, es una conexión dinámica, así que cada vez que me conecto a Internet el proveedor me asigna una IP diferente.

Es por ello que publico la IP, je je je.

Bien, ya tenemos la IP que el ISP nos ha asignado, pero ¿cómo averiguamos todas las IP del proveedor?.

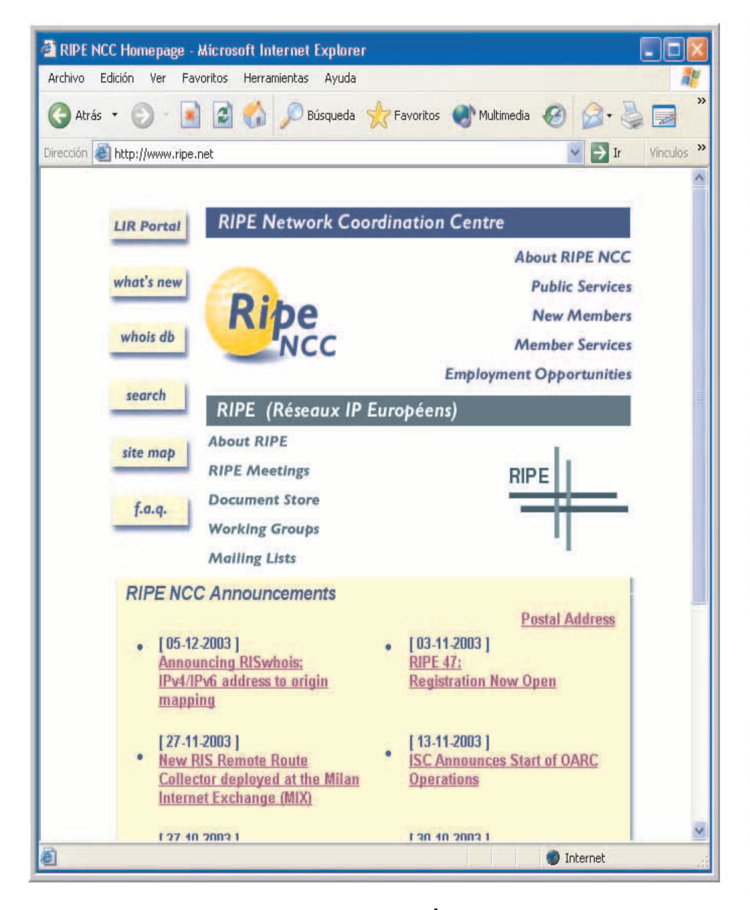
Ahora abrimos el navegador y ponemos la URL: http://www.ripe.net, guarda esta URL en tusfavoritos ya que te será de gran utilidad.

Ripe es uno de los cuatro Reginal Internet Registries que existen en el mundo. Ripe se encarga de controlar y de gestionar los recursos de Internet de Europa, África y Asia Central. Ripe mantiene la base de datos de los rangos de Ips y esta información es pública. Resumiendo, Ripe es el organismo que se encarga de controlar la distribución de los rangos IPS y que además nos permite consultar la base de datos.

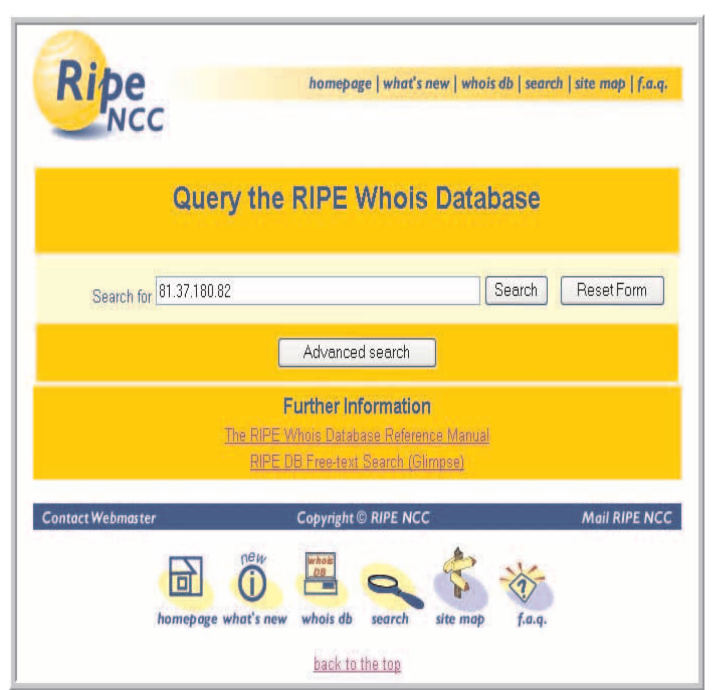


Un ISP...

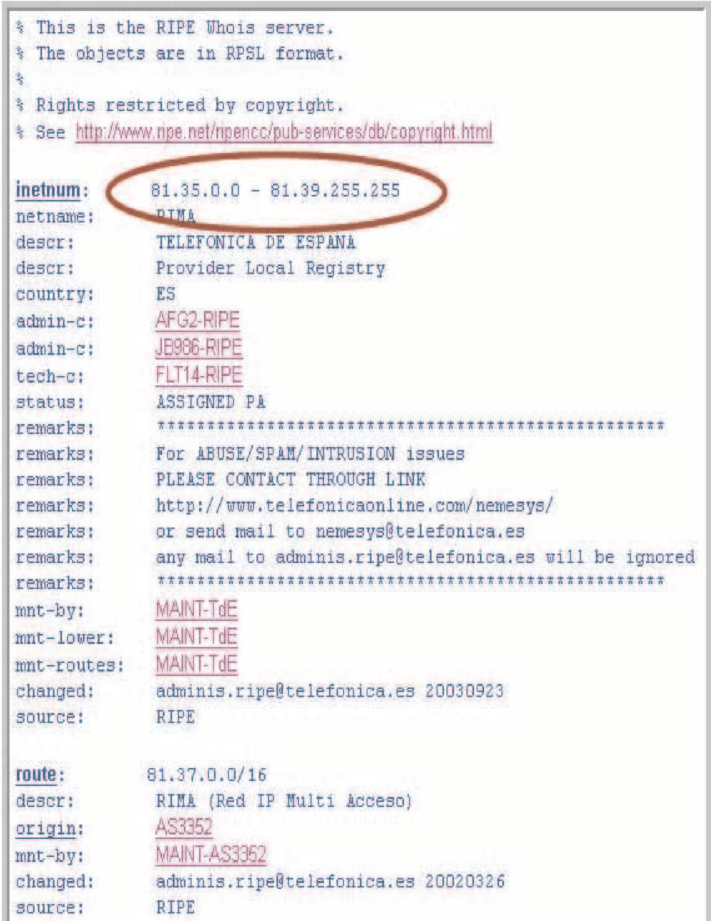
Un ISP, por ejemplo Telefónica, para darte una IP primero ha tenido que pedirle al RIPE que le ceda unas cuantas (un rango).



Pinchamos en la opción Whois db y colocamos la IP que nos ha asignado el proveedor.



A los pocos segundos ya tenemos la respuesta.



```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum:      81.35.0.0 - 81.39.255.255
netname:      RIMA
descr:        TELEFONICA DE ESPANA
descr:        Provider Local Registry
country:      ES
admin-c:      AFG2-RIPE
admin-c:      JB986-RIPE
tech-c:       FLT14-RIPE
status:       ASSIGNED PA
remarks:      ***************************************************
remarks:      For ABUSE/SPAM/INTRUSION issues
remarks:      PLEASE CONTACT THROUGH LINK
remarks:      http://www.telefonicaonline.com/nemesys/
remarks:      or send mail to nemesys@telefonica.es
remarks:      any mail to adminis.ripe@telefonica.es will be ignored
remarks:      ***************************************************
mnt-by:       MAINT-TdE
mnt-lower:    MAINT-TdE
mnt-routes:   MAINT-TdE
changed:      adminis.ripe@telefonica.es 20030923
source:       RIPE

route:        81.37.0.0/16
descr:        RIMA (Red IP Multi Acceso)
origin:       AS3352
mnt-by:       MAINT-AS3352
changed:      adminis.ripe@telefonica.es 20020326
source:       RIPE
```

El dato que buscamos se encuentra en **inetnum,** el whois nos muestra mucha más información interesante pero de momento nos quedamos con la información del **inetnum**.

Los que nos dice **inetnum** es que la IP introducida se encuentra dentro del rango 81.35.0.0 - 81.39.255.255 y que el propietario de dicho rango de IP es de Telefónica.

Ya que tenemos el rango de IP. Ahora nos hace falta crear el programa en PHP para que nos cree todas las IPs de ese rango. La primera comenzará por 81.35.0.0 y la última será 81.39.255.255.

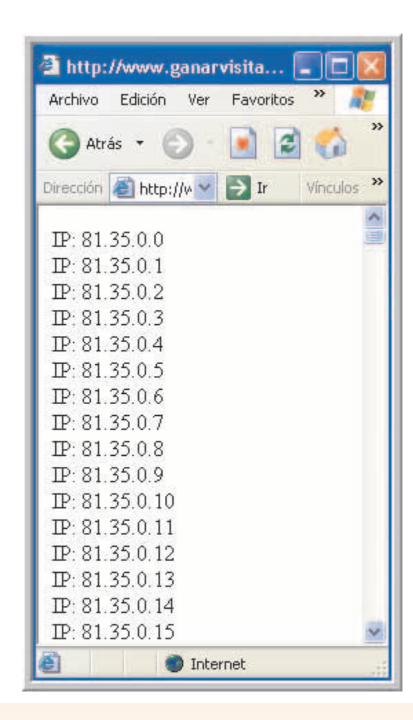
¿Serías capaz de hacer un programa en PHP que genere todas las IPS de ese rango?, inténtalo antes de ver la solución, con lo aprendido en

este capítulo tendrías que poder realizar el programa, aunque hay que reconocer que es algo complejo para comenzar.

```
<?php
$ipinicio[4]=81;
$ipinicio[3]=35;
$ipinicio[2]=0;
$ipinicio[1]=0;

$ipfin[4]=81;
$ipfin[3]=39;
$ipfin[2]=255;
$ipfin[1]=255;
$numero=0;
while (!(($ipinicio[4]==$ipfin[4]) && ($ipinicio[3]==$ipfin[3]) && ($ipinicio[2]=
                                        =$ipfin[2]) && ($ipinicio[1]==$ipfin[1]))) {
        $numero=$numero+1;
        print ("IP: $ipinicio[4].$ipinicio[3].$ipinicio[2].$ipinicio[1] <br>");
        $ipinicio[1]= $ipinicio[1]+1;
        if ($ipinicio[1]==256) {$ipinicio[1]=0; $ipinicio[2]=$ipinicio[2]+1;}
        if ($ipinicio[2]==256) {$ipinicio[2]=0; $ipinicio[3]=$ipinicio[3]+1;}
        if ($ipinicio[3]==256) {$ipinicio[3]=0; $ipinicio[4]=$ipinicio[4]+1;}
}
print ("Total: $numero");
?>
```

Ejecútalo y obtendrás la siguiente lista :)





Este código....

Este código deberías entenderlo perfectamente, pero si tienes dudas entra en el foro de HackxCrack y pregunta!!! (www.hackxcrack.com).

El programa se puede mejorar bastante, incluso existen funciones en PHP para realizar esta tarea, pero el objetivo del programa es practicar con lo aprendido.

Este ejemplo da como resultado 327679 IPs, una cantidad muy grande.

Si el tiempo de ejecución de la página supera los 60 segundos es posible que no termine de mostrar todas las Ips y de un error de TimeOut el PHP, para evitar que se corte la generación de IP hay que poner al comienzo del programa la sentencia: **set_time_limit($time_limit);**

Por ejemplo, si queremos que el script esté generando IPs durante una hora, hay que poner al comienzo del programa **set_time_limit(3600);** siempre en segundos.

El ejemplo anterior se podía haber simplificado utilizando las funciones ip2long() y long2ip().

Como ya se ha comentado una dirección IP consiste en un número de 32 bits, que se suele representar mediante 4 dígitos separados por puntos. Cada uno de estos dígitos están comprendidos entre 0 y 255. Para simplificar el ejemplo anterior, ¿no sería más sencillo pasar las IP a valores numéricos de 32 bits?, aquí entran en juego las funciones ip2long() y long2ip().

Si os fijáis para representar el numero 255 hacen falta 8 bits y como son cuatro números sería (4 * 8 = 32 bits).

Ip2Long()

Convierte una cadena cuyo contenido sea una dirección Ipv4 a valor numérico de 32 bits.

```
<?php
$ip=”81.35.0.0”;
print ip2long($ip); // da como resultado 1361248256
?>
```

Long2IP()

Es lo inverso de ip2long(), da como resultado una cadena con formato Ipv4 separado por puntos.

```
<?php
$ip= 1361248256;
print long2ip($ip);  // Da como resultado 81.35.0.0
?>
```

Con estas funciones podemos crear un pequeño programa que haga lo mismo que el anterior:

```
<?php
$ipinicio=ip2long("81.35.0.0");
$ipfinal=ip2long("81.39.255.255");
for ($ip=$ipinicio;$ip<=$ipfinal;$ip=$ip+1) {
        print  ("IP:".long2ip($ip)."<br>");
}
?>
```

El resultado es el mismo pero la ejecución es mucho más rápida.

En el próximo número...
Continuaremos con nuevas funciones PHP y con otro gran ejemplo hasta llegar a programar (no queda mucho) sockets, entonces si que empezaremos a jugar y a divertirnos. ;)

SERIE XBOX LIFE

INSTALANDO LINUX EN UNA XBOX

POR ALFONSO MENKEL

- Con esta serie de artículos aprenderemos a sacarle el máximo rendimiento a la consola XBOX
- Instalaremos LINUX en la XBOX
- Descubriremos el Disco Duro la consola de Microsoft

Bienvenidos al mundo XBOX; todos sabéis que XBOX es la consola más potente del mercado, pero seguro que no sabíais que con muy poca pasta y algo de maña la podemos convertir en un PC, y así sacarle el máximo partido a la consola.

Os podéis estar preguntado ¿por qué convertir la XBOX en un PC?, ¿qué ventajas tengo? Pues a mí se me ocurren unas cuantas, pero me quedo con ésta: "tendréis un ordenador más en casa, para hacer lo que queráis"

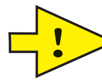
Primero quiero dejar claro, que no me hago responsable de cualquier daño físico o psicológico que pueda sufrir la consola, tú o tu mascota. Si sigues este articulo dentro de un margen razonable de investigación, no tiene porque pasar nada malo. Quedáis avisados.

1. Cosas que necesitamos.

Aquí os dejo una lista de las cosas que debemos tener para poder instalar Linux en nuestra XBOX (Todos los programas son para Windows).

- Consola XBOX + Mod chip.
-Cd-Rw (si puede ser de audio mejor).
- Ed's XBOX Debian GNU-Linux.
- PC con grabadora de Cd´s.
- 2 Cables conversor de USB XBOX
- Teclado y ratón USB
- Tarjeta de red en el PC
- Cable de red
- Nero 6.0 .0.19 ó Superior
- Win RAR
- ExtractNow 3.35

Consola XBOX + Mod Chip: No voy a explicar qué es ni como se instala el mod chip, hay mucha información en Google, buscad un poco que no es nada difícil. Aconsejo el Aladin Advance última versión, es barato, flasheable, y muy fácil de instalar. Pasaos por www.satkit.com



El Mod Chip...

El Mod Chip es un "añadido" a la consola que debe ser comprado e instalado. Normalmente requiere una serie de soldaduras y, si soldar no es tu fuerte, mejor dejas este tema en manos de profesionales.

El mes que viene veremos las "utilidades" de instalar un Mod Chip (que son muchas ;p) y le sacaremos el máximo partido a la XBOX.

Hay tiendas donde venden la XBOX con el Mod Chip ya instalado. Si estas a punto de comprarte esta consola, no lo dudes, COMPRALA CON EL Mod Chip ya instalado.

En Google puedes encontrar sitios donde te la venden de esta forma, por ejemplo http://www.artecnova.com /clientes/maxelectronica/chips.php

Según anuncian te recogen la consola en tu domicilio, te montan el chip y te la devuelven "preparadita" :)

Esta editorial no tiene ninguna relación con dicha empresa,

es un link que hemos sacado consultando el Google (www.google.com), hay muchos más y puedes elegir el que tu quieras :)

No te pierdas las próximas entregas de PC PASO A PASO. Te enseñaremos a sacarle el máximo partido a esta cónsola, descargar juegos de Internet, ver DIV-X (ficheros de video), emuladores... ...

Cd-Rw: Como algunos sabrán, Xbox no lee Cd´s normales (Cd-R), sólo lee algunos Cd-Rw, Cd´s de Audio o DVD´s, así que por lo menos uno de estos debéis tener a mano.

Ed's XBOX Debian GNU-Linux: Es la distribución de Linux que vamos a instalar en nuestra consola. Lo descargamos desde http://heanet.dl.sourceforge.net/sourceforge/xbox-linux/dist-1.0.0-beta2.tar.bz2 Es la versión más reciente en el momento de escribir este artículo.

PC con grabadora de Cd´s: Como es lógico el PC con grabadora de Cd´s es necesario.

2 Cables conversor de USB XBOX
Es para el uso de Linux.

No es totalmente necesario, pero sí aconsejable, y digo no es necesario porque hay otros 2 modos muy poco cómodos para manejar Linux en la consola, que explicaré más adelante. Sólo deciros que si os pasáis por www.satkit.com veréis que el cable + teclado USB vale 10€ + gastos de envío. Necesitaríais 2, uno para el teclado y otro para el ratón.

Teclado y ratón USB: Supongo que no necesito explicar para que son estas dos cosas tan raras ;-)

Tarjeta de red en el PC: Debemos tener instalada una tarjeta de red en nuestro ordenador. Para poder conectar desde la XBOX a Internet o al PC.

Cable de red: Para conectar la consola al PC, tiene que ser un cable JR 45, puede ser "cruzado" o no, depende como queráis hacer la conexión: directamente de la XBOX al PC ("cruzado") ó de la consola a un swich-hub, ... y después al PC ("no cruzado").

WinRar: Todos debéis conocer ya el programa WinRar, un compresor como el archiconocido WinZip. Con este programa descomprimiremos el dist-1.0.0-beta2.tar.bz2 (vamos, el Debian, es decir, el Linux).

Puedes descargar el WinRar desde http://winrar.com.es/

Nero 6.0.0.19 o Superior: Cualquier persona que alguna vez ha grabado un CDs debe conocer ya este programa. Lo usaremos para grabar Linux Debian.

Puedes comprarlo (www.nero.com) o descargarlo desde www.mocosoft.com y desde el emule ;p (tú eliges).

¿No sabes lo que es el emule? Pues pásate por http://www.spanishare.com o pregunta en el foro de esta revista (www.hackxcrack.com) y cientos de personas te informarán.

ExtractNow 3.35: Es un programa que nos permite extraer los archivos de las ISO (explicare lo que son más adelante) Es FREEWARE.

Descargad este programa desde:

http://download.com.com/3002-2250-10228183.html?tag=dir

2. Manos a la Obra.

Doy por hecho que ya tenéis el Mod chip puesto y funcionando.

Ahora nos ponemos a bajar la distribución de

Linux, desde el link que os he facilitado antes, son unos 261 Mb así que paciencia, iros a fumar si es que fumáis, y si no pues a comer algo que esto tardará un rato.

Ya lo tenemos bajado en nuestro disco duro (que a partir de ahora lo llamaré HD -Hard Drive-) del ordenador.

Ahora instalamos los programas anteriormente mencionados, tan sencillo como cualquier instalación en Windows, no hace falta ni explicarlo.

Con el Win-Rar descomprimimos el Archivo dist-1.0.0-beta2.tar.bz2 a una carpeta en nuestro HD, por ejemplo en

C:\xbox-linux\

El resultado de esta descompresión será este archivo: dist-1.0.0-beta2.tar, pues lo volvemos a descomprimir en la misma carpeta, nos quedaran 2 carpetas y 13 archivos + el *.tar + el *.bz2, de los cuales sólo los siguientes son necesarios para la instalación:

- 1.0.0-beta2.ISO Cd Instalación
- boot_fatx_e.ISO Cd Arranque E:\
- boot_hdd.ISO Cd Arranque F:\



Antes de seguir...

Antes de seguir una nota aclaratoria:

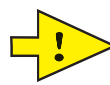
No he hablado del famoso Evolution X (evox, evo-x) por una simple razón, se sale del tema que hoy nos trae aquí, así que si buscáis en google encontraréis la información necesaria sobre ello. No obstante el mes que viene lo veremos con detalle ;)

Para los que lo tienen instalado o tenéis cosas ya en el HD (disco duro) de la consola, os aviso que vayáis haciendo hueco/copia de seguridad de las unidades E:\ y F:\. Para los que no lo tienen y no han tocado el HD de la consola, que no se preocupen, son los que más fácil lo tienen.

No sé si lo sabéis, pero la XBOX tiene varias particiones en su HD (¿Cómo que no sabías que la XBOX tiene HD? ¿Pero en que mundo vives?)

Las particiones que podemos tocar sin miedo a estropear nada son: E:\ y F:\ .

E:\ es donde se guardan las partidas de los juegos y las canciones de los CD´s que hemos pasado al HD, tiene unos 5 Gb de espacio libre.



En los proximos...

En los próximos artículos veremos cómo ampliar el HD de la consola, además de otras muchas cosas.

F:\ en cambio no tiene formato, además es posible que ni tengas partición F:\ ya que en algunas consolas el HD no viene con los 2 Gb extra.

Si no sabes si tiene o no la partición F:\ tendrás que instalar Linux en la partición E:\. Los demás debéis decidir donde lo queréis instalar. Si lo instaláis en la partición F:\ Perderéis todo lo que haya dentro, en cambio si lo hacéis en E:\ no perderéis nada.

Si miramos el HD de la consola y vemos que es de la marca SEAGATE, entonces eso quiere decir que tenemos 2 GB extra, si no es de esta marca, no.

3. Grabando con Nero

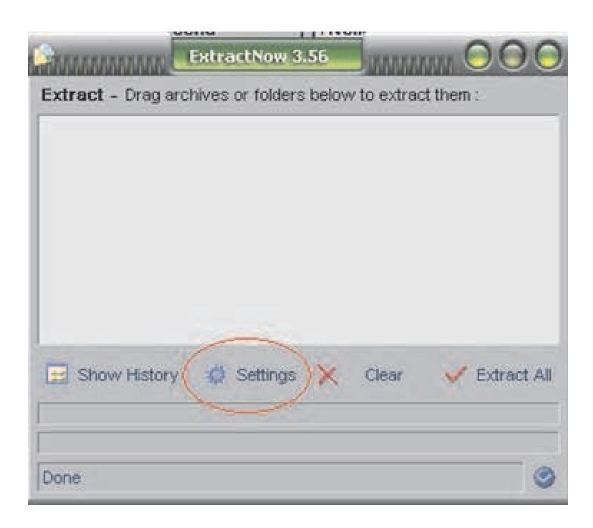
Espero que tengáis ya todo lo que os he dicho que os bajéis.

Vamos a tratar con las ISO, seguro que si tenéis una grabadora sabréis lo que son, pero de todas formas voy a dar una breve explicación.

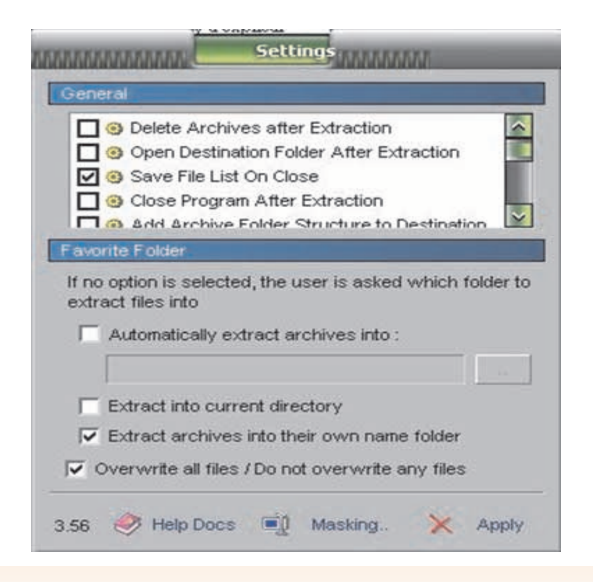
Las ISO son archivos que contienen otros archivos, como un archivo comprimido, pues esto es lo mismo, pero con una variante, también guarda información de cómo deben ser grabados adecuadamente tales archivos.

Ya sé que la explicación no es muy buena, pero creo que será suficiente por ahora.

Arrancamos el ExtractNow 3.35:

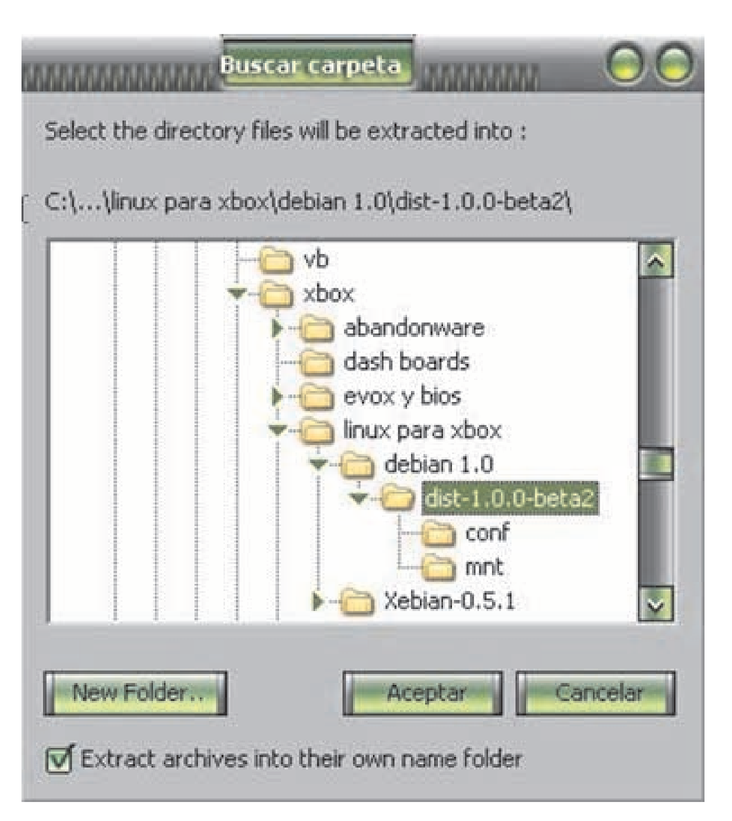


Pinchamos en Settings para configurar el programa:



Como no queremos que nos extraiga la ISO en la misma carpeta donde la tenemos, pues deseleccionamos la casilla "EXTRACT INTO CURRENT DIRECTORY" y pinchamos en "APPLY".

Arrastramos el 1.0.0-beta2.ISO dentro del ExtractNow 3.35 y pinchamos en "EXTRACT ALL"



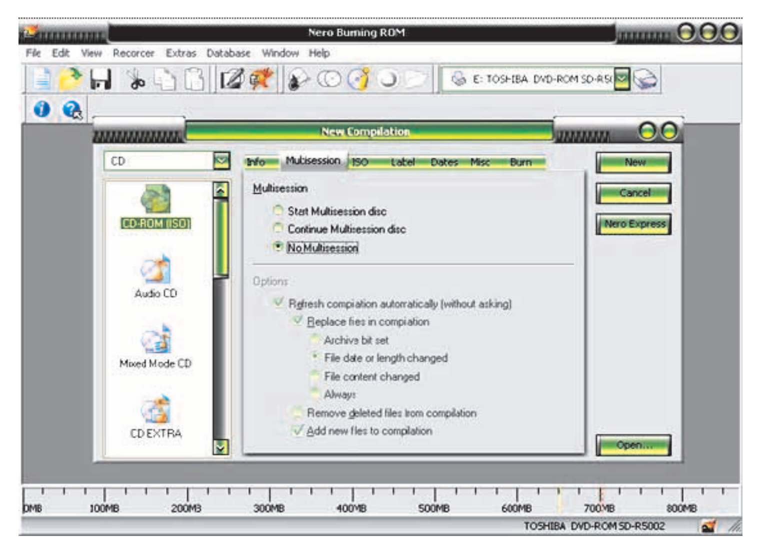
Seleccionamos el destino de los archivos de la ISO y pinchamos en "ACEPTAR"

¿Por qué extraemos la ISO si esta contiene la información necesaria para grabar adecuadamente los archivos?

Muy sencillo, el tal ED no sabe grabar los archivos adecuadamente para XBOX, así que tenemos que extraerlo y grabarlo bien y así estar seguros de que no nos va a fallar.

Mientras se extrae la ISO vamos instalando el NERO....

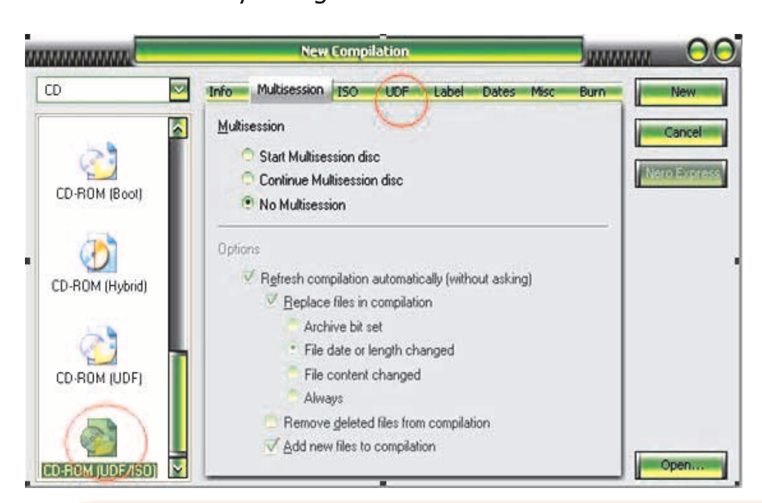
¿Ya está? Pues arrancad el NERO:



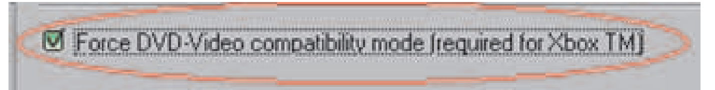
Si os sale la pantalla de "WIZZARD" o el "NERO EXPRESS" debéis pasar a modo clásico. (La imagen de arriba).



Seleccionamos CD ó DVD depende de donde lo vayáis a grabar.



Seleccionamos "UDF/ISO" abajo del todo y pinchamos en la pestaña UDF y marcaremos la siguiente casilla:

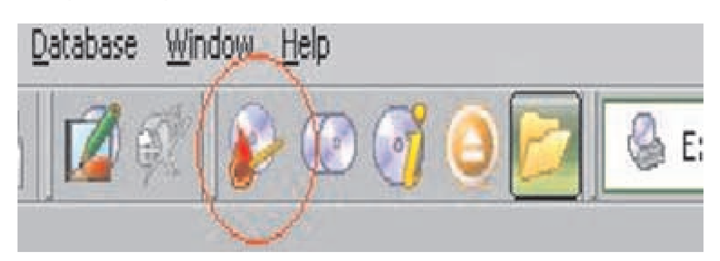


OJO!!! MUY IMPORTANTE. Esta opción no la encontraréis en las versiones anteriores de NERO.

Ahora pinchamos en Label y ponemos un nombre al CD, por ejemplo LINUX.

Pinchamos en NEW.

En el lado derecho de la ventana vemos el contenido de nuestro HD, vamos a la carpeta donde están los archivos que extrajimos de la ISO, las seleccionamos todas y las arrastramos a la parte izquierda de la ventana.



Pinchamos en la cerilla de la barra de menú, nos saldrá otra pantalla, nos aseguramos que la opción de Finalizar CD esté activada, pincháis en "BURN" y a esperar.

No os he hablado de las otras opciones del NERO porque hay que dejarlas tal cual están por defecto.

4. Instalando:

Por fin tenemos el CD grabado; lo metemos en la consola.

El led cambiará a naranja. Es normal.

Esto es lo primero que veremos:



Y después de que se cargue, saldrá esta:



Como veis tenemos un teclado virtual, es un infierno usar este método, pero si no tenéis los cables USB para conectar un teclado de verdad o no podéis esperar a que os lleguen, usad el virtual.

El otro método es usar el teclado del PC a través de la red, pero también es un fastidio si no tienes la consola cerca, así que este método no lo explicaré.

El nombre de usuario que debes introducir es root y la contraseña xbox, luego cuando esté instalado podemos cambiar la contraseña.

Después de autentificarnos (introducir el nombre de usuario y la contraseña), tendremos una pantalla típica de Linux.

Tendréis el XCONSOLE abierto.
A mí me ha dado problemas, así que he abierto el terminal (el icono de la pantalla, en la parte inferior Izquierda).

Para empezar la instalación debemos convertirnos en súper-usuario, para hacerlo teclearíamos esto:
su –

Ahora arrancamos el instalador poniendo:

XBOXLinuxInstall

Recordad que en Linux hay que respetar las mayúsculas y minúsculas.

Ahora nos pide dónde queremos instalar Linux.

Como ya os lo expliqué antes, ya debéis saber donde lo queréis instalar, yo he elegido E:\ porque mi HD no viene con los 2 GB extra. ;-(

Así que he elegido la primera opción:

"Install XBOX LINUX INSIDE THE GAME PARTITION"

Si elegís la partición E:\ os saldrá un mensaje preguntando si queremos preparar el HD para instalar Linux, al que responderemos: USE.

Si no lo instaláis en E:\ no sé realmente lo que sucede, pero supongo que no os lo pedirá, sino que lo hará sin daros opción.

Ahora nos pregunta el tamaño que queremos que tenga la partición Swap, por defecto está en 256 MB; yo lo he dejado así, tened en cuenta

que si no habéis ampliado el HD, la consola sólo tiene unos 4,9 GB, pero podéis poner lo que queráis, siempre que la suma de la partición Swap y ROOT no supere los 4,8 GB.

Ahora nos pide el tamaño de la partición ROOT, está puesto por defecto en 2 GB, es el máximo, así que si no vais a poner una cantidad menor, no lo toquéis.

Lo que ahora hace Linux es instalar y crear unos archivos que van a simular un HD (imagen de disco duro), así que no os tenéis que preocupar, ni tener miedo de fastidiar la configuración de la consola, ya que cuando uséis Linux no vais a estar manejando el HD de la consola, si no una imagen de tal.

Esperamos a que se instale, esto tardará un rato largo, así que mientras tanto vamos a por el segundo CD que tenemos que grabar (si sólo tenéis 1 CD-RW este paso lo haréis después de la instalación).

Extraemos y grabamos tal y como he explicado antes la ISO correspondiente a donde habéis instalado Linux, es decir que si habéis instalado en E:\ extraéis y grabáis boot_fatx_e.ISO y si Fuera en F:\ tendríais que usar el boot_hdd.ISO

Ahora que ya ha acabado la instalación, debemos configurar la conexión de red.

Por defecto en Linux viene esta:

IP: 192.168.0.2
NETMASK:255.255.255.0

Configurarla según vuestra red de Windows. La configuración por defecto en LINUX será válida si vuestra IP de la tarjeta de red en Windows es el 192.168.0.1.

¿Que no sabéis qué IP local tenéis en Windows? Pues deberíais saberlo, se ha explicado muchas veces en la revista :) y si no, preguntad en el foro (www.hackxcrack.com).

Bueno, venga, lo explico muy rápido, para saber la IP tan solo hay que abrir una Ventana de Comandos en Windows (Menu Inicio --> Programas --> Accesorios --> Símbolo del sistema) y poner "ipconfig /all" y os saldrá la IP.

Debéis poner como IP de Linux una mayor que la que tenéis en el PC.

En Gateway: deberéis poner la IP de vuestra tarjeta de red del PC, en mi caso 192.168.0.1

Host Name: El nombre de host de Linux es por defecto XBOX.

Lo demás como está.

Ya está configurada la red, ahora lo que vamos a hacer es salir y apagar la consola.

Sacamos el CD y si ya habéis grabado el segundo lo metéis, y si no pues borramos el de instalación y grabamos el de arranque.

No explico el borrado de un CD-RW porque me parece que no es necesario, preguntad en el foro si tenéis dudas.

5. Arrancando:

Conectamos los cables de la red. Si con lo anteriormente expuesto no os funciona la Red, preguntad en el foro.

Metemos el CD-RW en la consola y se cargará Linux….

Por fin ya tenemos Linux instalado; nos autentificamos como root.
Nos vamos al navegador Mozilla y ponemos como web: www.hackxcrack.com



Ya está hecho, solo queda cambiar la contraseña del root.

Abrimos una shell y escribimos: passwd
Nos pedirá la contraseña antigua, lo ponemos (xbox) y luego 2 veces seguidas la nueva contraseña.

Como el root tiene acceso a todo, para lo bueno y lo malo creo que sería conveniente crear otro usuario.

Aunque ya hay uno:
live/live.

Para agregar otro usuario:
En la shell escribimos: adduser, rellenamos lo que nos pide y a disfrutar del nuevo PC.

Por último deciros que si compráis un VGABOX (unos 40€) y lo conectáis al monitor lo veréis mucho mejor.

El mes que viene veremos qué es y como se instala el EvolutionX y muchas cosas más ;p

¡¡¡¡¡Feliz año a todos!!!!!!!!

RAW 9
MSN (MICROSOFT MESSENGER)

- Descubriremos el Protocolo MSN (MESSENGER)
- Estudiaremos los Servidores que intervienen en un Inicio de Sesión en MSN.
- Nos conectaremos por TELNET y nos autentificaremos por MD5
- Chat y FTP en MSN

1. INTRODUCCION

Me he resistido bastante a escribir este artículo ya que, al margen de mi manía particular hacia Microsoft, en general quería centrar la serie RAW en estándares públicos, y no en protocolos propietarios de una compañía, como es el caso del protocolo que utiliza MSN. Pero desde el principio supe que tarde o temprano tendría que escribir sobre este tema, ya que es sin duda uno de los que más pueden interesar a los lectores en general.

Yo, personalmente, no soy usuario de MSN, pero como hay que conocer siempre al "enemigo", considero igualmente interesante conocer el protocolo que utiliza.

Como ya he comentado, este protocolo no es un estándar público, si no que es propietario de Microsoft. Por tanto, no existe ningún RFC que detalle su funcionamiento. Aún así, hay información bastante completa que podéis encontrar fácilmente a través de Google y, por supuesto, siempre nos queda la ingeniería inversa. ;-)

Antes de empezar, he de decir que existen gran cantidad de protocolos diferentes, ya que con cada nueva versión de MSN el protocolo se ha ido complicando. Por tanto, no puedo garantizar que lo que explique aquí tenga que funcionar necesariamente, y menos aún que lo que explique aquí sea TODO el protocolo. Por tanto, tomad este artículo más como un primer acercamiento a este protocolo, que como una especificación técnica detallada.

Si queréis información completa y al día sobre este protocolo, tenéis una fuente muy buena en: http://www.hypothetic.org/docs/msn/index.php.

2. ARQUITECTURA DE MSN

El protocolo de MSN, como cualquier otro de los explicados hasta ahora en la serie RAW, funciona a base de comandos que son enviados desde un cliente hacia un servidor a través de una conexión TCP. El puerto utilizado en este caso es el **1863**. Una diferencia notable entre este protocolo y otros, como el FTP, o el HTTP, es que el servidor no sólo envía al cliente respuestas a los comandos que éste solicita, si no que además puede enviar mensajes propios al cliente en cualquier momento. Esto es lógico si pensamos por ejemplo que los usuarios de tu lista de contactos pueden conectarse y desconectarse en cualquier momento, y el servidor ha de avisarte cuando esto ocurra, aunque tú no lo estés solicitando constantemente.

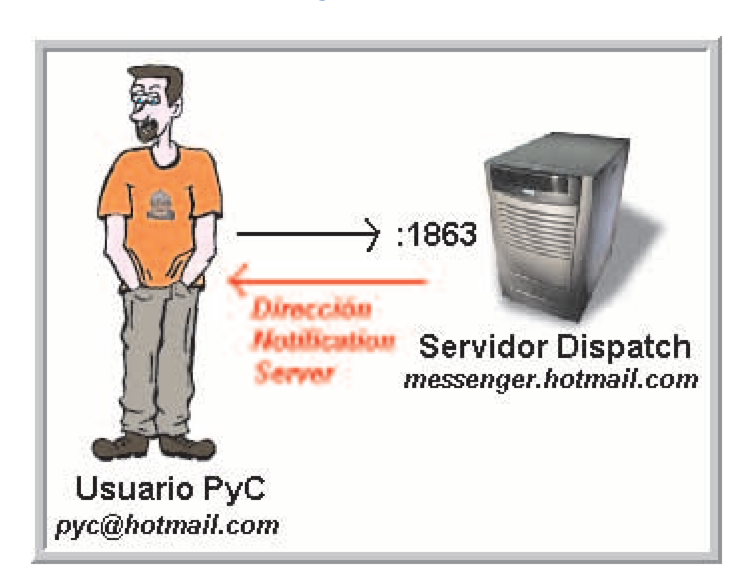
En este sentido, y en otros tantos, hay muchos conceptos comunes entre IRC y MSN, aunque en realidad ambos protocolos distan mucho, tanto en su gramática, como en su arquitectura, y en cuestiones de diseño.

La cuestión más importante de la arquitectura de MSN es la organización de los **servidores**,

ya que puede parecer un poco confusa al principio.

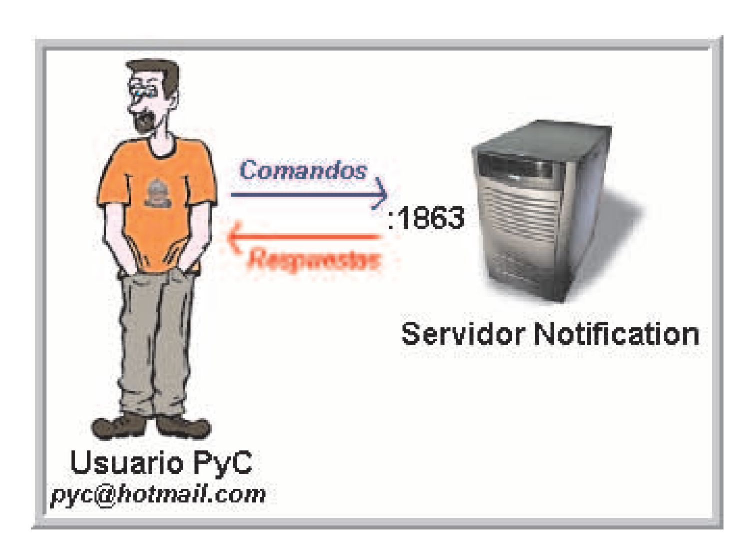
Una sesión de MSN no es tan simple como en la mayoría de los protocolos, en los cuales simplemente se establece una conexión TCP entre nuestro cliente y un servidor y ale, a tirar comandos. En el caso de MSN, hay 3 tipos de servidores diferentes con los que podemos establecer conexiones a lo largo de una única sesión.

La sesión comienza cuando establecemos una conexión con el puerto 1863 de un servidor único, que es el llamado **Dispatch**, cuya url es messenger.hotmail.com.



Una vez conectados con el servidor **Dispatch**, le pedimos que nos diga con qué servidor podemos conectarnos para llevar a cabo el resto de la sesión. Por tanto, el servidor **Dispatch** sólo sirve para repartir los recursos (servidores) disponibles entre todos los usuarios. El servidor **Dispatch** nos dará la dirección del servidor que debemos utilizar, el cual se llama dentro de esta arquitectura **Notification Server**.

A continuación, cerramos la conexión con el servidor **Dispatch**, para realizar una nueva conexión con el servidor **Notification** cuya IP nos acaba de facilitar el **Dispatch**.



El resto de la sesión se llevará a cabo en el servidor **Notification**, que será el que reciba nuestros comandos, y nos envíe las respuestas, así como los mensajes asíncronos que deba enviarnos (como los avisos de que un usuario de tu lista se ha conectado o desconectado).

El tercer tipo de servidor, llamado **SwitchBoard**, sólo entrará en juego si establecemos un Chat, para lo cual debemos hacer una solicitud al servidor **Notification**, el cual nos proporcionará la dirección de un servidor **SwitchBoard** que podremos utilizar exclusivamente para esa sesión de Chat. El resto de comandos seguirán gestionándose a través del servidor **Notification**, y podremos solicitar nuevas sesiones de Chat, lo que dará lugar a una conexión con un nuevo servidor **SwitchBoard** (Vease IMAGEN A en la página siguiente)

En la ilustración podemos ver una sesión de ejemplo, en la que PyC tiene establecida una sesión de chat con el usuario Scherzo, y otra con la usuaria Adhara (¿a que está bien la chica? ;-).

3. FORMATO DE LOS COMANDOS

Antes de empezar a mostrar los comandos, tenemos que conocer unos cuantos aspectos

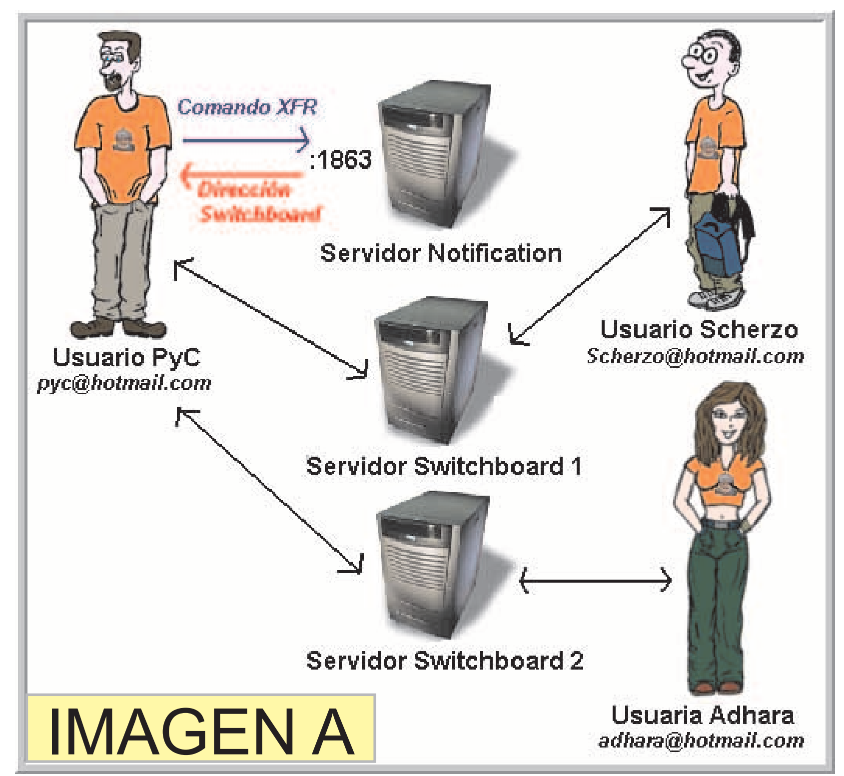


IMAGEN A

comunes a todos ellos.

Todos los comandos empiezan por un código identificador de 3 caracteres. Este código indica el tipo de comando y, aunque hay muchos comandos diferentes, se puede hablar de 3 tipos especialmente importantes:

- **Mensajes:** Su código identificador es **MSG**, y son los que se usan para transmitir mensajes de texto, como por ejemplo cuando estás hablando en un chat.
- **Errores:** Su código es un **número de 3 cifras**, que codifica el tipo concreto de error. Son enviados siempre del servidor al cliente. A lo largo del artículo iremos viendo algunos de los errores más importantes.
- **Resto de comandos:** Cualquier otra combinación de 3 letras reconocida por el protocolo. Si probamos alguna combinación que no corresponda con ningún comando, el servidor nos responderá con un **error 200 (Syntax Error).**

Después del identificador de tipo de comando, siempre vendrán uno o más parámetros, separados por espacios. Por tanto, si queremos incluir espacios dentro de un parámetro, tendremos que codificarlo mediante **urlencode**. Tenéis una especificación completa de este sistema de codificación en el **RFC 1738**, pero en realidad muchas veces basta con saber simplemente que el **espacio** se codifica con el ya conocido **%20** (puedes repasar los números 1, 2 y 3 de PC PASO A PASO / Los Cuadernos de Hack x Crack donde se explicó extensamente "eso del %20").

El único sistema de codificación que se usa no es urlencode, que se utiliza sólo para los parámetros, pues también se usa el sistema **UTF-8**, pero en este caso para el cuerpo de los mensajes de texto (**MSG**). Tenéis la especificación completa de UTF-8 en el **RFC 2279**. Igual que os resumía antes el urlencode, con respecto a UTF-8 basta decir que mientras no uséis caracteres raros (es decir, prácticamente cualquier cosa que no sean letras sin tildes, y números) no tenéis que hacer nada, ya que este sistema sólo sirve para codificar caracteres que se salgan del estándar ascii de 127 caracteres.

¿RFC?...

¿RFC? ¿%20? ¿urlencode? ¿UTF-8? Argggg… ¿de qué estamos hablando?
En los números anteriores se ha explicado extensamente todo lo relativo a los RFCs y sus traducciones al español. Si eres un nuevo lector debe sonarte a chino, no desesperes, en www.hackxcrack.com puedes pedir los números atrasados.

Por último, sólo queda hablar de un parámetro fundamental, que se encuentra como primer parámetro en todos los comandos, incluidos

los errores (de hecho, en los errores, es el único parámetro), y exceptuando únicamente los mensajes (MSG). Este parámetro lo encontrábamos también precisamente en mi último artículo, que trataba sobre el protocolo DNS, y tiene el mismo significado en este caso. Se trata del **Transaction ID**, o identificador de transacción (**TRID**). Es un número único que identifica una determinada transacción, es decir, una combinación de solicitud+respuesta. Cada vez que el cliente envíe un comando al servidor, lo hará con un TRID diferente, y el servidor deberá responder a ese comando con el mismo TRID. Más que una medida de seguridad se trata de una medida de gestión, ya que permite distinguir qué respuesta corresponde a qué comando cuando hemos solicitado varios comandos al servidor y estamos esperando varias respuestas.

Digo que no es muy práctico como medida de seguridad porque los clientes de MSN suelen utilizar TRIDs secuenciales, es decir, empiezan por el TRID 0 y a cada nuevo comando van usando el próximo TRID. El TRID es un número de 32 bits, por lo que tendríamos que enviar más de 4 mil millones de comandos en una sola sesión para que hubiera problemas potenciales con el TRID.

Igual que el TRID sirve para saber qué respuesta corresponde a qué comando, por supuesto, también sirve para saber qué error corresponde a qué comando.

El servidor también puede utilizar nuevos TRIDs cuando envíe al cliente mensajes asíncronos (por ejemplo, cuando un usuario de la lista de contactos se conecte y tenga que notificárselo al cliente).

4. COMENZANDO LA SESIÓN

Una vez calentados los motores de nuestra aplicación de **Telnet:**

telnet messenger.hotmail.com 1863

Con eso realizamos el primer paso para comenzar una sesión, que es conectar con el servidor **Dispatch**.

En realidad, no os aconsejo que probéis este protocolo mediante Telnet, ya que los comandos son muy largos y muy pesados de escribir, y además el servidor os puede desconectar fácilmente si no recibe lo esperado en el momento esperado (ante lo cual posiblemente os responderá con un **error 715**). Por tanto, en mi opinión la verdadera utilidad de este artículo está en conocer el protocolo bien para programar vuestras propias aplicaciones que exploten algún aspecto concreto del protocolo, o bien para alguna ocasión en la que necesitéis saber lo que está ocurriendo en vuestra sesión de MSN mediante un **sniffer**.



Uffff...

Ufff... si no sabes utilizar TELNET nunca has leído esta revista, te aconsejamos que revises los números anteriores. TELNET te permite conectarte a un servidor remoto DIRECTAMENTE y "hablar" con él (enviarle comandos), en PC PASO A PASO hemos explicado muchas veces su funcionamiento y es una de nuestras herramientas de trabajo más "básicas" :)

Una vez conectados, vamos a lanzar ya nuestro primer comando, el comando VER, que sirve para especificar la versión del protocolo a utilizar. Este es el comando:

VER 0 MSNP7 MSNP6 MSNP5 MSNP4 CVR0

Como ya dijimos, los 3 primeros caracteres identifican el tipo de comando, en este caso el comando VER (Version), y el primer parámetro es siempre el TRID que, en este caso, al ser el primer comando que enviamos, es el 0. El resto de parámetros son específicos del comando

VER, ya que el único parámetro obligatorio en la mayoría de los comandos es el TRID.

Por supuesto, antes de hacer nada más, lo primero es identificarnos. Para ello empezamos consultando el sistema de autenticación que debemos utilizar:

INF 1

A este comando nos responderá el servidor con el tipo de desafío que se utilizará para la autenticación. Actualmente sólo se usa el sistema **MD5**, que lo tenéis definido en el RFC 1321. Pero tranquilos, que aún no hemos de autenticarnos. Esto se hará posteriormente, en el servidor **Notification**.

De momento, lo único que nos interesa es que el Dispatch nos diga con qué servidor **Notification** debemos conectar para el resto de la sesión.

Suponiendo que nuestra cuenta en hotmail sea pyc@hotmail.com, el siguiente paso será identificarnos mediante este comando: **USR 2 MD5 I pyc@hotmail.com**

Después del TRID, como segundo parámetro, repetimos el sistema de autenticación que nos indicó el servidor que debíamos utilizar (aunque aún no lo vamos a utilizar). El tercer parámetro, la I, indica que este comando Inicia el proceso de autenticación, ya que este proceso se realiza en varios pasos, y éste será el primero. El último parámetro, por supuesto, es nuestra cuenta en Hotmail, que es lo que nos identifica.

A este comando el servidor nos debería responder con algo como esto:

XFR 2 NS 207.46.106.120:1863 0 207.46.104.20:1863

Lo que nos interesa de todo esto es el tercer parámetro: 207.46.106.120:1863. Ésta es la IP y el puerto que hemos de utilizar para conectar con el Notification Server. Ya podemos cerrar la conexión con el Dispatch, y abrir una nueva con el Notification:

telnet 207.46.106.120 1863

Con esto conectamos con el servidor Notification. Una vez aquí, tenemos que repetir los pasos anteriores, para luego continuar:

VER 3 MSNP7 MSNP6 MSNP5 MSNP4 CVR0
VER 3 MSNP7 MSNP6 MSNP5 MSNP4 CVR0
INF 4
INF 4 MD5
USR 5 MD5 I pyc@hotmail.com

Esta vez, el servidor ya no nos responderá con un XFR, si no que nos solicitará que continuemos el proceso de autenticación. Para ello, nos enviará la cadena aleatoria para el desafío MD5:

USR 5 MD5 S 1443856346.37654

Como vemos, esta vez no se trata de una I, si no de una S, ya que no es el primer paso de la autenticación. Se sale del tema de este artículo explicar en detalle la autenticación MD5, así que os recomiendo que simplemente utilicéis una aplicación que haga el trabajo por vosotros. Tenéis una en http://www.fourmilab.ch/md5/

A partir de la cadena aleatoria que nos ha proporcionado el servidor, construimos el md5sum (mediante una aplicación como la que he mencionado) de la cadena resultante de concatenar el número aleatorio con nuestra password. Si, por ejemplo, nuestra password es: LCo, tendremos que hacer el md5sum de esta cadena: **1443856346.37654LCo**

Para el caso concreto de la aplicación que os he mencionado, ejecutaríamos desde una ventana MS-DOS o desde una shell de Linux/Unix lo siguiente:
md5 -d1443856346.37654LCo –l

A lo cual la aplicación nos daría la siguiente cadena md5sum:
776e48f7a65f1ae6f3420d3520d55d9e

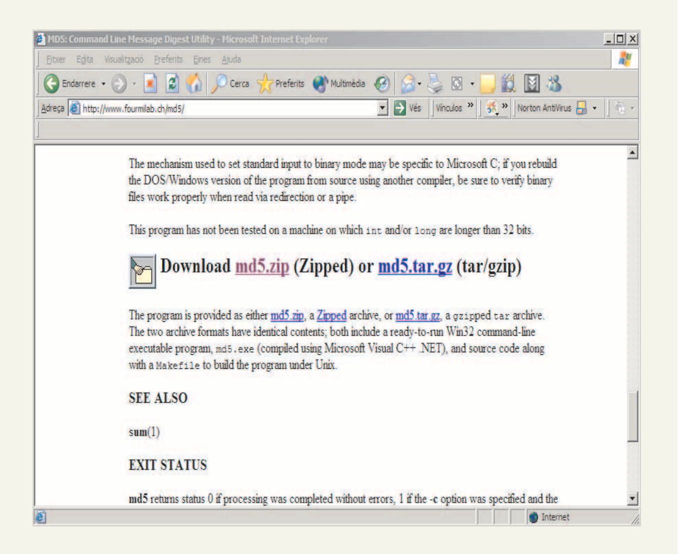


OYE!!!...

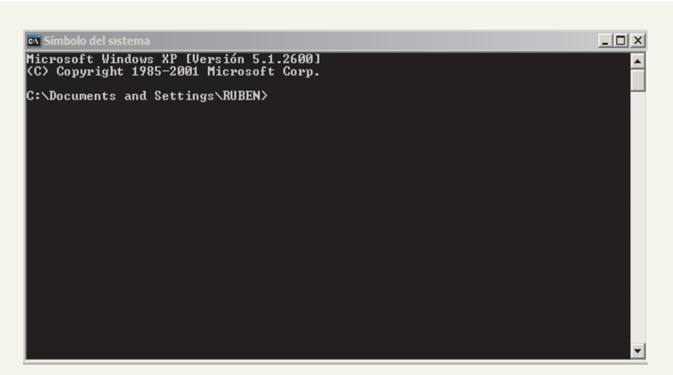
- OYE!!! Que yo pico con el ratón sobre el md5.exe y me sale una ventana negra que no funciona… ¿Cómo se hace el md5sum de la cadena "144385646.37654LCo"?

Vale, vamos a explicarlo paso a paso para aquellos que nunca nos han leído y posiblemente esta sea una de las últimas veces que lo haremos.

1.- Vamos a la WEB que hemos mencionado anteriormente http://www.fourmilab.ch/md5/ y bajamos un poco hasta encontrar el enlace para descargar el programa (puedes verlo en la imagen de color rojo). Pulsamos sobre él y guardamos el archivo donde queramos (nosotros lo dejamos en C:\)



2.- Descomprimimos el archivo en la carpeta c:\md5
3.- Iniciamos una Ventana de Comandos, algo explicado una y mil veces, por ejemplo, en Windows XP Vamos al Menu Inicio --> Ejecutar y en la ventanita que nos saldrá escribimos cmd.exe y aceptamos (otra forma de hacerlo, vamos al Menu Inicio --> Programas --> Accesorios --> Símbolo del Sistema). Ahora tendremos una "preciosa "ventanita negra" como la de la imagen.



4.- Ahora ejecutamos el programa md5.exe (que si has seguido nuestros pasos estará en C:\md5\) con la cadena que queremos "descifrar" y las opciones pertinentes. Pues venga, escribimos la siguiente instrucción:

```
c:\md5\md5.exe -d1443856346.37654LCo -l
```

… después pulsaremos enter y obtendremos la cadena md5sum que estabamos buscando (776e48f7a65f1ae6f3420d3520d55d9e). Puedes verlo en la pantalla.

```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\RUBEN>c:\md5\md5.exe -d1443856346.37654LCo -l
776e48f7a65f1ae6f3420d3520d55d9e

C:\Documents and Settings\RUBEN>
```

Para los que están muy despistados:
- c:\md5\ --> es la ruta, es decir, donde hemos descomprimido el programa que nos hemos bajado.
- md5.exe --> es el programa que queremos ejecutar
- -d1443856346.37654LCo -l --> es la cadena que deseamos "traducir" con las opciones -d al principio y -l al final (si quieres saber más sobre las opciones de este programa lee la documentación del mismo, que nos quedamos sin espacio en la revista).
- finalmente obtenemos el resultado: 776e48f7a65f1ae6f3420d3520d55d9e

Muchos pensarán que a estas alturas esta explicación tan detallada es totalmente innecesaria, pues en Hack x Crack estamos totalmente de acuerdo, pero después nos llegan mails preguntando cosas tan sencillas como estas y nos volvemos locos. En la semana del 15 al 21 de Diciembre inauguramos "nueva WEB" y esperamos poder poner en

ella este tipo de explicaciones tan básicas en lugar de publicarlas en la revista. Venga, os dejamos con el resto del artículo :)

Ya podemos enviar nuestra respuesta al desafío del servidor mediante Telnet:

USR 6 MD5 S 776e48f7a65f1ae6f3420d3520d55d9e

Si todo ha ido bien, el servidor nos responderá con algo como esto:

USR 6 OK pyc@hotmail.com PyC 1

Donde **PyC** es nuestro nick en MSN, y el **1** indica que nuestra cuenta ha sido verificada. Teniendo en cuenta que los passwords se codifican con MD5, y que se utiliza cada vez una cadena aleatoria diferente, mejor será que os vayáis olvidando de robar cuentas de MSN con un sniffer. ;-)

El siguiente paso será indicarle al servidor que nuestro usuario ha cambiado de **estado**, ya que pasa de estar en estado desconectado a estar **online**, para lo cual ejecutamos el comando:

CHG 7 NLN

Que indica que nuestro nuevo estado es **NLN (Online).** El servidor difundirá este cambio de estado a los usuarios que te tengan en su lista de contactos.

En lugar de conectar directamente como NLN, podríamos conectar como **invisibles** con el estado **HDN (Hidden):**

CHG 7 HDN

En este caso, el servidor no difundiría nada, y para el resto de usuarios aparentemente seguirías desconectado.

En cualquier momento, podremos volver a cambiar nuestro estado mediante el comando CHG, siendo estos los estados más importantes:

NLN (oNLiNe : conectado)
FLN (oFfLiNe : desconectado)
HDN (HiDdeN : conectado, pero aparece como desconectado para el resto de usuarios)
IDL (IDLe : el usuario lleva inactivo cierto tiempo)
AWY (AWaY : ausente)
BSY (BuSY : ocupado, no molestar) 

A lo largo de la sesión no sólo podemos cambiar nuestro estado, si no también nuestro nick. Para ello utilizaremos el comando **REA**:

REA 8 pyc@hotmail.com PyC%20LCo

Con esto hemos cambiando nuestro nick a **PyC LCo**, por supuesto, codificado con urlencode (%20 en lugar del espacio). El problema es que Microsoft impone una censura sobre los nicks, pero hay formas de saltarse esto. ;-)
Si codificamos todo el nick, y no sólo los espacios y caracteres especiales, podrá pasar los filtros de Microsoft. :-D

En este momento, podemos enviar un comando para identificar el software que estamos utilizando, y su versión, así como el sistema operativo, pero este comando no es obligatorio. Si queremos hacerlo:

CVR 8 0x0409 win 4.10 i386 MSMSGS 4.5.0127 MSMSGS

Con esto indicamos que nuestro sistema operativo es **Windows 98** en un procesador **Intel 386**, y que nuestro cliente es la **versión 4.5 build 127 de Microsoft Messenger**. El servidor nos responderá con el número de versión actual, y la URL desde la que podemos bajarla:

CVR 8 4.5.0127 4.5.0127 1.0.0863 http://download.microsoft.com/download/msnmessenger/install/4.5/win98me/en-us/ mmssetup.exe http://messenger.microsoft.com

En cualquier momento podemos desconectar con el comando **OUT**:

OUT

5. UNA VEZ CONECTADOS...

Por nuestra parte, ya está todo hecho para comenzar la sesión (aunque no ha sido fácil). Ahora es tarea del servidor (que trabaje un poco) enviarnos toda la información que necesitamos cada vez que conectamos.

En primer lugar, nos enviará dos mensajes de **Hotmail**, dándonos información sobre nuestra cuenta, y sobre los mensajes que tenemos en nuestra bandeja de entrada. Estos dos mensajes comenzarán con:

MSG Hotmail Hotmail ...

Pero lo único que puede parecernos interesante de todo esto es el campo **Inbox-Unread**, que contiene el número de mensajes sin leer que tenemos en la bandeja de entrada.

En cualquier momento a lo largo de nuestra sesión, nos pueden llegar nuevos mensajes de Hotmail para indicarnos que hemos recibido un **email**.

Lo que sí que es más interesante, es la notificación del estado de cada usuario de nuestra **lista de contactos**. Para ello, el servidor utilizará un nuevo TRID en el cual nos enviará, línea por línea, el estado de cada usuario. Por ejemplo:

ILN 9 NLN Scherzo@hotmail.com Scherzo
ILN 9 AWY Adhara@hotmail.com Adhara
ILN 9 BSY skitter@hotmail.com skitter

El servidor nos indica aquí que el usuario **Scherzo** se encuentra **conectado (NLN),** que el usuario **Adhara** está **ausente (AWY),** y que el usuario **skitter** está **ocupado (BSY).**

Los usuarios que estén desconectados no aparecerán en esta lista.

En cualquier momento, el estado de un usuario puede cambiar, y en este caso el servidor no utilizará el comando **ILN**, si no bien el comando **NLN** o el **FLN**. El **NLN** se usará cuando el nuevo estado no sea desconectado (o**NL**i**N**e, **AW**a**Y**, o **B**u**SY**). El **FLN** se usará cuando el nuevo estado sea desconectado. Por ejemplo:

NLN AWY Scherzo@hotmail.com Scherzo

Nos indica que el usuario **Scherzo** ha cambiado su estado a **ocupado (AWY).** Mientras que:

FLN Scherzo@hotmail.com Scherzo

Nos indica que el usuario **Scherzo** se ha **desconectado.**

6. LISTAS DE CONTACTOS

Aunque en un principio pueda parecer que existe una única lista de contactos, en realidad hay que tener en cuenta varias listas.

En primer lugar, la mas obvia, es la lista de usuarios que hemos ido añadiendo para conocer su estado y comunicarnos con ellos. Pero también es importante la lista contraria, que es la de aquellos usuarios que te han añadido a ti a su lista. Con estos usuarios también puedes establecer una comunicación.

Además de estas dos listas, hay dos listas especiales, la de Allow y la de Block, que sirven para indicar qué nivel de privacidad deseamos tener frente a qué usuarios.

Por último, hay que recordar también que la lista de contactos que hemos añadido nosotros no es única, ya que podemos agruparla en distintas listas. Por ejemplo, una lista para tus

amigotes, otra para los compañeros de clase/trabajo, otra para la familia, etc.

Por tanto, podemos manejar un número indeterminado de listas, aunque las más importantes son las 4 primeras que nombré:

• **FL (Forward List)** : Tu propia lista de contactos, incluidos los de todos los grupos.

• **RL (Reverse List)** : La lista de los usuarios que te tienen en su propia FL.

• **AL (Allow List)** : Lista de usuarios para los que deseas ser visible.

• **BL (Block List)** : Lista de usuarios para los que no deseas ser visible.

Por su puesto, las listas **AL** y **BL** son mutuamente excluyentes.

Cada lista tiene un **número de versión** que se incrementa cada vez que se hace una modificación sobre esa lista (se añaden o se eliminan usuarios). El número de versión de cada lista es mantenido por el servidor.

Si queremos que el servidor nos dé una de nuestras listas, la solicitamos con el comando **LST**. Por ejemplo, para solicitar nuestra lista **FL**:

LST trid FL

Donde trid es el identificador de transacción que corresponda utilizar en ese momento. A esto nos responderá el servidor con algo parecido a esto:

LST trid 100 1 4 Scherzo@hotmail.com Scherzo 0
LST trid 100 2 4 Adhara@hotmail.com Adhara 0
LST trid 100 3 4 skitter@hotmail.com skitter 1
LST trid 100 4 4 pollodios@hotmail.com Pollo%20Dios 2

Vamos a analizar esta respuesta:

En primer lugar, podemos deducir fácilmente que el servidor nos envía una línea de respuesta por cada usuario que tengamos en esa lista (en este caso, la lista **FL**).

El primer parámetro, después del TRID, es el número de versión de la lista FL. A continuación tenemos un número que indica la posición dentro de la lista, y después otro que indica el número total de elementos en la lista. A continuación, el email y el nick. Y por último, un número que indica el **grupo** en el que tenemos a ese usuario. Por ejemplo, en este caso podríamos tener un grupo, el grupo 0, para los miembros de LCo, otro grupo, el 1, para los compañeros de la facultad, y otro grupo, el 2, para los Pollos.

Podemos ver que en nuestra FL hay un usuario, **Pollo Dios**, que no nos fue enviado por el servidor en un comando **ILN** cuando nos conectamos. Esto se debe a que este usuario no estaba conectado en el momento de conectarnos nosotros (o bien estaba invisible).

Podemos gestionar estos grupos mediante comandos. Por ejemplo, con el comando:

ADG trid Familia 0

Añadimos el grupo familia a nuestra FL. El servidor nos responderá con la nueva versión de la lista FL, y con el código que corresponde a ese grupo:

ADG trid 101 Familia 3

Es decir, la lista FL está ahora en su versión **101**, y al grupo **Familia** le corresponde el **código 3.** Podemos borrar un grupo con:

RMG trid 3

Con esto borraríamos el recién creado grupo 3, es decir, el grupo Familia.

Con respecto a la gestión de usuarios dentro de una lista, disponemos de los comandos **ADD** y **REM** para añadir y borrar respectivamente. Así:

ADD trid FL soulnet@hotmail.com Soulnet 0

Con esto añadimos al usuario **Soulnet** al **grupo LCo (grupo 0).**

Para borrar al usuario recién añadido:

REM trid FL soulnet@hotmail.com

Si, en lugar de añadir a la lista **FL, añadimos a las listas AL** o **BL**, los efectos son diferentes. Si añadimos un usuario a nuestra lista **AL**:

ADD trif AL soulnet@hotmail.com Soulnet

Indicamos al servidor que permitimos que ese usuario pueda conocer nuestro estado. En este caso, no hay que indicar un código de grupo, ya que no hay grupos dentro de la lista AL, sólo en la lista FL.

Si, en cambio, queremos bloquear a un usuario, tendremos que añadirlo a nuestra lista **BL**. En caso de que estuviese previamente añadido a nuestra lista AL, tendremos que eliminarlo primero, ya que si no el servidor nos devolverá un **error 219**:

REM trid AL soulnet@hotmail.com
ADD trid+1 BL soulnet@hotmail.com Soulnet

Podemos obligar a que un usuario reciba nuestra **confirmación** si quiere añadirnos a su FL (es decir, si quiere añadirse a nuestra RL), mediante el comando:

GTC trid N

Si en lugar de **N** el parámetro fuese **A**, indicaríamos lo contrario: que cualquier usuario podría añadirnos a su lista sin pedirnos permiso.

En el momento en que un usuario nos añada a su lista, el servidor nos avisará de que se ha añadido ese usuario a nuestra **RL**, con un mensaje como este:

ADD 0 RL 120 soulnet@hotmail.com Soulnet

Con esto nos indica que el usuario Soulnet nos ha añadido a su FL y, por tanto, se ha añadido a la versión 120 de nuestra RL. Cuando ocurra esto, decidiremos qué hacer con este usuario, añadiéndolo en nuestra AL si queremos que sea nuestro amiguito, o en la BL si no queremos saber nada de él (o, más bien, que él no sepa nada de nosotros).
Por supuesto, el servidor también nos avisará de las bajas en nuestra RL, con un mensaje como este:

REM 0 RL 121 soulnet@hotmail.com

Que nos indica que el usuario Soulnet nos ha borrado de su lista FL.

7. CHAT

Como ya dijimos, para las sesiones de chat se utiliza un tercer servidor, que es el **SwitchBoard**. Si queremos iniciar una sesión de chat, lo primero será solicitar la dirección de un SwitchBoard que esté disponible. Para ello utilizamos el siguiente comando:

XFR trid SB

La respuesta del servidor:

XFR trid SB 64.4.12.193:1863 CKI 12398789312.541378664.23146

No sólo nos indica la **IP** y **puerto** del servidor **SwitchBoard**, si no que también nos da una **cadena aleatoria** que necesitaremos para la

autenticación en el servidor.

A continuación, por supuesto, lanzamos el nuevo **Telnet**, sin cerrar el anterior, ya que seguiremos necesitando la conexión con el Notification Server para todos los demás eventos de la sesión

telnet 64.4.12.193 1863

Dentro del **SwitchBoard** comenzamos identificándonos:

USR 1 pyc@hotmail.com 12398789312.541378664.23146

Como veis, repetimos la cadena que nos dio el Notification Server. Una vez conectados e identificados, podemos **invitar** a otros usuarios a que se unan al Chat:

CAL 2 Scherzo@hotmail.com

Con esto, invitamos al usuario Scherzo a que se una a nuestra sesión de chat en el SwitchBoard. El servidor nos responderá:

CAL 2 RINGING 2893765

Indicándonos el identificador de sesión que dará a ese usuario.

El servidor avisará a Scherzo con un mensaje como este:

RNG 2893765 64.4.12.193:1863 CKI 34482642.436429847 pyc@hotmail.com PyC

Que le indica quién le ha solicitado que se una al chat (**PyC**), en qué servidor (**64.4.12.193** en el puerto **1863**), la cadena de identificación (**34482642.436429847**), así como el identificador de sesión (**2893765**).

Si ahora él quiere unirse al chat, deberá empezar por conectar con el **SwitchBoard**:

telnet 64.4.12.193 1863

Para, a continuación, identificarse para comenzar la sesión:

ANS 1 Scherzo@hotmail.com 34482642.436429847 2893765

Como vemos, repetimos los dos números que nos fueron dados en el comando **RNG**. Si la identificación es satisfactoria, el servidor nos dará la lista de usuarios conectados al chat:

IRO 1 1 3 pyc@hotmail.com PyC
IRO 1 2 3 Adhara@hotmail.com Adhara
IRO 1 3 3 soulnet@hotmail.com Soulnet
ANS 1 OK

Como vemos, el formato es similar al de la respuesta al comando **LST**.

Si Scherzo aceptó nuestra invitación, y realizó todo el proceso descrito anteriormente, recibiremos un mensaje como este:

JOI Scherzo@hotmail.com Scherzo

Indicándonos que Scherzo se ha unido a nuestro chat.

La conversación en el chat, desde el punto de vista del protocolo RAW, no es trivial, ya que se utiliza el estándar **MIME**, del cual ya he hablado en otros artículos de la serie (sobre todo en el que trataba sobre el protocolo HTTP). Por tanto, cada texto enviado al chat ha de ir acompañado de una cabecera MIME aunque, por suerte, bastante sencilla en comparación con cómo puede llegar a complicarse una cabecera MIME.

Además, el protocolo MSN permite especificar el nivel de confirmación que deseamos que nos de el servidor por cada mensaje enviado:

- **U :** El servidor no envía ningún tipo de confirmación. Nosotros vamos escribiendo, y no sabemos si los interlocutores están leyendo lo que escribimos.

• **N :** Por cada mensaje que escribimos, el servidor nos enviará una línea con un comando **NAK trid** si el mensaje no llegó a su destino.

• **A :** Por cada mensaje que escribimos, el servidor no sólo nos responderá si el mensaje no llegó (**NAK trid**), si no que también nos confirmará en caso de que llegase correctamente (**ACK trid**)

Asumiendo que no deseamos ningún tipo de confirmación (**U**), éste sería un mensaje de ejemplo:

```
MSG trid U 149
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
X-MMS-IM-Format: FN=Arial; EF=; CO=000000; CS=0; PF=22

Hola!! :-DDD
```

El último parámetro del comando (**149**) indica el **tamaño en bytes** del mensaje, incluida la cabecera.

Como vemos, indicamos que utilizamos codificación **UTF-8**, que ya dijimos que es la que se utiliza para los mensajes de texto.

La línea **X-MMS-IM-Format** indica varios parámetros sobre el tipo de letra utilizado. Para una escritura normal podéis copiar directamente esa línea.

Nuestros interlocutores en el chat recibirán un mensaje del servidor como este:

```
MSG pyc@hotmail.com PyC 149
MIME-Version: 1.0
Content-Type: text/plain; charset=UTF-8
X-MMS-IM-Format: FN=Arial; EF=; CO=0; CS=0; PF=22

Hola!! :-DDD
```

Podemos, además, enviar un mensaje especial que simplemente indica al resto de interlocutores que nos encontramos en ese momento escribiendo un mensaje (desde luego, si lo hacemos por telnet, no tiene mucho sentido).

El mensaje sería similar a este:

```
MSG U 90
MIME-Version: 1.0
Content-Type: text/x-msmsgscontrol
TypingUser: pyc@hotmail.com
```

Lo curioso de este tipo de mensaje es que podemos escribir cualquier cosa en el campo **TypingUser**, por lo que podríamos engañar al resto de usuarios, haciéndoles creer que es otro usuario el que está escribiendo.

Por último, para finalizar la sesión de chat (es decir, cerrar la conexión con el **SwitchBoard**) podemos usar el mismo comando que cierra la conexión con el Notification Server:

```
OUT
```

Ante lo cual, el resto de interlocutores recibirán un mensaje como este:

```
BYE pyc@hotmail.com
```

Indicándoles que el usuario PyC ha abandonado el chat.

8. ENVIANDO ARCHIVOS

El equivalente al **DCC** en el protocolo MSN (**MSNFTP**) dista mucho en realidad del mismo, no sólo porque la sintaxis es diferente, si no también porque aquí hay una mayor seguridad, al haber un diálogo entre ambas partes de la transferencia que permite asegurarse hasta cierto punto de que estás enviando el archivo al usuario que deseas, y no a otro cualquiera que se ha colado en la conexión, como veíamos en el caso de los protocolos DCC y FTP en anteriores artículos. Antes de que tiréis cohetes en favor de Microsoft, tened en cuenta que los protocolos DCC y FTP existen desde la era de los dinosaurios, y el protocolo MSNFTP sólo

desde el año 98, así que ya sería delito, a esas alturas, no haber diseñado un protocolo medianamente seguro.

En lo que sí que se parecen DCC y MSNFTP es en que la conexión se realiza punto a punto entre ambos usuarios, actuando, normalmente, el usuario transmisor como servidor, y el receptor como cliente.
Vamos a ver detenidamente los pasos que se siguen desde que el usuario PyC desea enviar un archivo al usuario Scherzo, hasta que éste lo recibe.

8.1. PyC envía a Scherzo una invitación para recibir un archivo.

Para ello, envía al servidor un mensaje como este:

MSG trid U 273
MIME-Version: 1.0
Content-Type: text/x-msmsgsinvite; charset=UTF-8

Application-Name: File Transfer
Application-GUID: {5D3E02AB-6190-11d3-BBBB-00C04F795683}
Invitation-Command: INVITE
Invitation-Cookie: 22692
Application-File: LCo.jpg
Application-FileSize: 57565

Este mensaje indica que queremos enviar un archivo llamado **LCo.jpg**, de tamaño **57565 Bytes**.

El parámetro más raro que podemos ver aquí es el llamado **Application-GUID**. Éste numerajo hexadecimal identifica unívocamente un tipo de aplicación, en este caso, la aplicación de transferencia de ficheros. Si utilizamos valores diferentes para el Application-GUID podemos construir invitaciones para servicios muy diferentes.

Por ejemplo, con:

Application-GUID: {02D3C01F-BF30-4825-A83A-DE7AF41648AA}

Se trataría de una invitación para un **chat por voz**.

No bastaría con modificar el **Application-GUID**, ya que el chat por voz, y cualquier otro tipo de invitación, tiene diferentes campos en la cabecera, así que lo pongo sólo como ejemplo.

En el caso que nos ocupa, que es el de la transferencia de ficheros, podríamos incluir un campo opcional, que sería:

Connectivity: N

Este campo indica que no tenemos posibilidad de abrir conexiones entrantes (probablemente porque estamos detrás de un firewall), así que invitamos al receptor del archivo a que sea él el que abra el puerto para la transferencia. En el resto del ejemplo, asumiremos que es PyC (el que ha hecho la invitación) el que abre el puerto, tal y como veremos más adelante.

8.2. Scherzo acepta la invitación de PyC

En el caso de que Scherzo desee recibir el archivo, enviará un mensaje con el formato siguiente:

MSG pyc@hotmail.com PyC 179
MIME-Version: 1.0
Content-Type: text/x-msmsgsinvite; charset=UTF-8

Invitation-Command: ACCEPT
Invitation-Cookie: 22692
Launch-Application: FALSE
Request-Data: IP-Address:

Como vemos, en este mensaje Scherzo acepta la invitación de PyC, y además le solicita los

datos necesarios para la conexión: la IP y el puerto al que debe conectarse. En el caso de que fuese Scherzo el que tuviese que actuar como servidor (si PyC hubiese enviado el parámetro **Connectiviy: N**) este mensaje sería totalmente diferente, pero no entraremos en detalles, ya que sólo hablamos del caso más general.

Si, en cambio, Scherzo no aceptase la invitación, podría rechazarla con este mensaje:

MSG pyc@hotmail.com PyC 146
MIME-Version: 1.0
Content-Type: text/x-msmsgsinvite; charset=UTF-8

Invitation-Command: CANCEL
Invitation-Cookie: 22692
Cancel-Code: REJECT

8.3. PyC facilita a Scherzo los datos necesarios para la conexión

A continuación, PyC tiene que dar a Scherzo los datos que necesita para establecer la conexión. Para ello, envía un mensaje como éste:

MSG trid N 239
MIME-Version: 1.0
Content-Type: text/x-msmsgsinvite; charset=UTF-8

Invitation-Command: ACCEPT
Invitation-Cookie: 22692
IP-Address: 217.20.11.135
Port: 6891
AuthCookie: 73856
Launch-Application: FALSE
Request-Data: IP-Address

Es fácil interpretar el significado de los parámetros. La **IP** y el **puerto** se encuentran en los parámetros **IP-Address**, y **Port**, respectivamente. El único parámetro que requiere una explicación es el **AuthCookie**. Este parámetro servirá para que luego el cliente se identifique a la hora de establecer la conexión. Esto es precisamente lo que diferencia el protocolo MSNFTP de otros protocolos de transferencia de archivos, como DCC, o FTP, en los cuales cualquiera puede conectarse a un puerto en escucha del servidor, y suplantar así la personalidad del cliente legítimo. Aquí no basta sólo con conectarse el primero, si no que además hay que demostrar que fue a nosotros a quien se hizo la invitación para la transferencia. Pero todo esto lo veremos ahora, en los siguientes pasos.

8.4. Scherzo se conecta a PyC

Ahora que el cliente de Scherzo conoce la IP y puerto que ha abierto PyC para la transferencia, ya puede establecer una nueva conexión TCP. En este caso:

telnet 217.20.11.135 6891

Por supuesto, si PyC está detrás de un **firewall**, tendrá que abrir el tráfico entrante para ese puerto. MSNFTP suele usar el puerto **6891** y los sucesivos (6892, 6893, ...), así que tendría que abrir estos puertos en el **NAT** del firewall en caso de encontrarse en una red privada, como es por ejemplo el caso de un usuario de ADSL con Router.

A partir de ahora, el resto de pasos se llevan a cabo dentro de esta nueva conexión establecida.

8.5. Scherzo se identifica, y comienza la transferencia

En el caso de DCC o FTP, llegados a este punto comenzaría automáticamente el flujo de datos nada más establecerse la conexión entre el cliente y el servidor. En cambio, aquí son

necesarias unas gestiones previas para identificar al cliente.

El primer paso es simple:

VER MSNFTP

A lo cual nos responderá con el mismo mensaje el servidor, indicando así que ambos están de acuerdo con el protocolo a utilizar: **MSNFTP**.

A continuación, viene el paso crucial, que es la identificación del cliente:

USR Scherzo@hotmail.com 73856

Como vemos, indica su nombre de usuario (Scherzo@hotmail.com), así como la **AuthCookie** que le dio el servidor antes de establecer la conexión y que, teóricamente, solo conocen ambas partes. Por supuesto, si con un **sniffer** has capturado el AuthCookie de otro usuario, podrías colarte tú en su lugar, y conseguir así un **Hijacking** como los que expliqué en los protocolos **DCC** y **FTP**. Si la autenticación ha sido correcta, el servidor nos responderá con el **tamaño del archivo en Bytes**:

FIL 57565

Después de esto, ya sólo falta que el cliente inicie la transferencia con el comando:

TFR

8.6. PyC transfiere el archivo a Scherzo

Como a los muchachos de Microsoft les gusta complicar las cosas, la transferencia del archivo no consiste en un simple flujo de datos, como ocurre en el caso de DCC o FTP.

El propio protocolo **TCP/IP** ya se encarga de empaquetar los datos y fragmentarlos, por lo que estos protocolos delegan por completo en los niveles inferiores (nivel de transporte: TCP, y nivel de red: IP) para estas tareas. En cambio, MSNFTP implementa su propio sistema de empaquetamiento de los datos en **bloques**, que es relativamente complejo.

Cada bloque tiene una pequeña cabecera de 3 Bytes, que especifica el tamaño del bloque (por defecto de **2045 Bytes**), de la siguiente manera:

- El primer byte de la cabecera, es siempre **0**, excepto en el último bloque, que es **1**.

- El segundo byte y el tercer byte especifican la **longitud del bloque** según la fórmula: (tercer byte) * 256 + (segundo byte)

Por tanto, el **último bloque**, que finalizaría la transferencia, constaría únicamente de los 3 bytes de cabecera, con el siguiente valor:

Primer byte = 1
Segundo byte = 0
Tercer byte = 0

Cuando Scherzo haya recibido todo el archivo, cerrará la sesión con el comando:

BYE 16777989

Ese número es un código que indica que la operación terminó con éxito. Existen otros códigos, como por ejemplo:

- **2147942405** : Disco del receptor lleno. No se puede continuar la transferencia.

- **2164261682** : El receptor ha cancelado la transferencia.

En lugar de utilizar estos códigos para cancelar la transferencia, se puede utilizar directamente el comando:

CCL

Que tiene el mismo efecto, en cualquier momento durante la sesión de transferencia.

Tanto con un **BYE** como con un **CCL** se cierra la conexión, con lo que nuestra atención volvería a las otras conexiones que tendremos (con un Notification Server, y con uno o varios SwitchBoards), y volvería a repetirse toda la historia que ya os he contado, por lo que aquítermina mi trabajo por el momento. ;-)

Autor: PyC (LCo)

CURSO DE VISUAL BASIC
UN CLIENTE, UNA NECESIDAD, TENEMOS UN PROYECTO (PARTE III)
POR PEDRO DEL VALLE

- Vamos a realizar unas cuantas mejoras al proyecto que tenemos entre manos.
- Vamos a acabar el Proyecto.

Hola de nuevo. Si no recuerdo mal, en el último número os "reté" a que hicierais por vosotros mismos el formulario de ventas del proyecto. Antes de esto, me gustaría depurar algunos errores que se pueden producir en el formulario de entradas, en algunos casos especiales.

DEPURANDO EL CÓDIGO...

El primero caso que puede provocar un error es cuando no tenemos registros en la lista. Si intentamos abrir la pantalla de entradas, sin que no tengamos ningún registro en ella, se produce un error que nos podría incluso cerrar la aplicación. Propongo para arreglarlo, dos métodos distintos. El primero, un control de errores en la función de "MoverDAtosACampos".

En esta rutina podríamos aplicar este control de tal manera que, en caso que se produzca un error, mostremos un mensaje que diga, por ejemplo, "La tabla de entradas está vacía"

Esto sería así:

```
Sub MoverDatosACampos()
    On Error GoTo Errores
    Txtnombre = Lista.SelectedItem.SubItems(1)
    TxtSabor = Lista.SelectedItem.SubItems(2)
    TxtCantidad = Lista.SelectedItem.SubItems(3)
    TxtFecha = Lista.SelectedItem.SubItems(4)
   Exit sub
Errores:
    MsgBox "La tabla de entradas está vacía", vbExclamation, "Aviso"
End Sub
```

Con este método le estamos diciendo al código que, si se produce un error en esta rutina, el proceso de un salto hacia la etiqueta Errores. Posteriormente mostramos el mensaje de aviso. Yo, personalmente, prefiero otro método, que paso a explicar. Cuando la tabla es abierta, podríamos comprobar si el valor del EOF del Recordset es True. Sería así:

```
Private Sub Form_Load()
    Set Rs = New ADODB.Recordset
    Rs.Open "entradas", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
    If Not Rs.EOF Then
        While Not Rs.EOF
            With Lista.ListItems.Add(, , Rs("Id"))
                .SubItems(1) = Rs("Nombre")
                .SubItems(2) = Rs("Sabor")
                .SubItems(3) = Rs("Cantidad")
                .SubItems(4) = Rs("Fecha")
            End With
            Rs.MoveNext
        Wend
        MoverDatosACampos
    Else
        MsgBox "La tabla de entradas está vacía", vbExclamation, "Aviso"
        CmdModificar.Enabled = False
        CmdBorrar.Enabled = False
    End If
    Deshabilitar
End Sub
```

Con este método comprobaríamos antes si hay o no registros en la tabla, y en caso de que no hubiera, no llamaríamos a la rutina "MoverDatosACampos".

Otro error, no tan importante, se produce a la hora de recargar la lista. En el botón aceptar hacemos una recarga

de la lista cuando se intenta interactuar contra algún registro que ha sido borrado o cambiado sin que se haya recargado la lista. Para ello, llamamos al evento Load del formulario. El problema es que no se limpia la pantalla cuando recargamos. Lo podemos solucionar limpiando antes los Ítem de la lista.

```
Private Sub Form_Load()
  Set Rs = New ADODB.Recordset
  Rs.Open "entradas", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
  Lista.ListItems.Clear
  If Not Rs.EOF Then
    While Not Rs.EOF
      ………
      ………
```

Para acabar, dos pequeños errores más. El primero, a la hora de modificar un registro. Si no tenemos ninguno en la tabla, no debería ni intentarlo. Lo solucionamos así:

```
Private Sub CmdAceptar_Click()
  Select Case Estado
    Case "MODIFICAR"
      If CamposOK = True And Not Rs.EOF Then
        Rs.MoveFirst
        Rs.Find "Id=" & Lista.SelectedItem.Text
          ………
          ………
```

Lo único que hemos cambiado es la sentencia condicional. Tenemos que evitar entrar dentro del "If" en caso que no haya registros.

El otro error lo encontramos en los bucles de recorrido "for", que en vez de hacerlo hasta el Listitems.Count –1 lo hacemos hasta el ListItems.Count.

```
For i = 1 To Lista.ListItems.Count
```

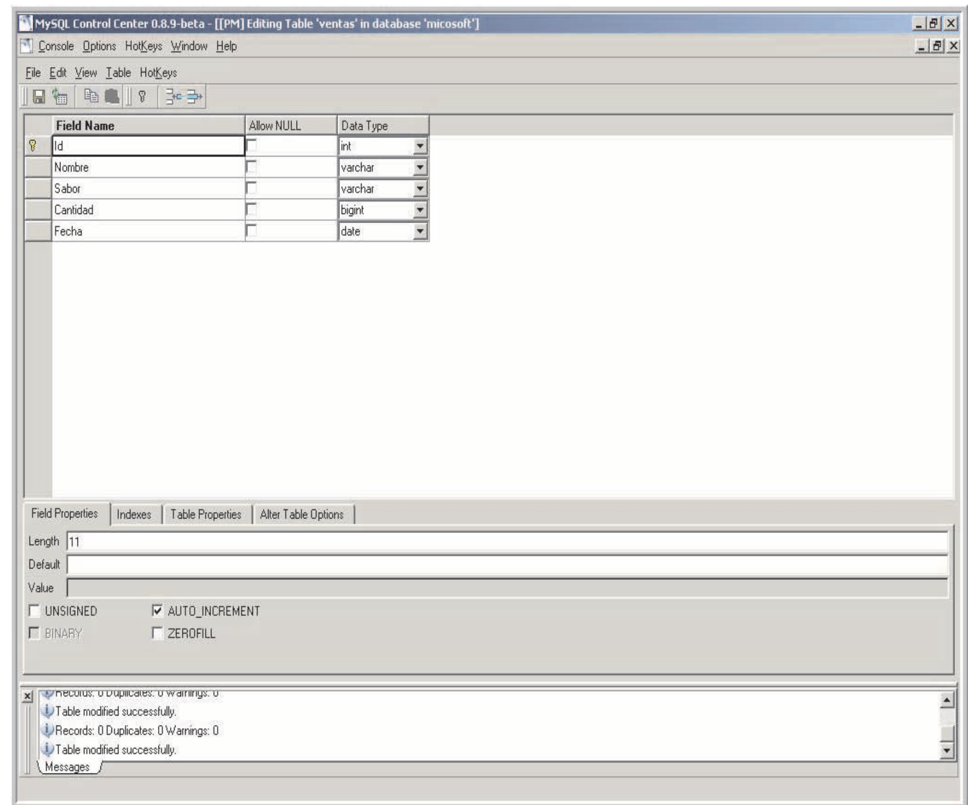
Bien, creo que con esto damos por finalizadas las correcciones por hoy, si encontráis alguna más, podéis comunicármelo en los foros de hackxcrack.

SEGUIMOS CON EL PROYECTO:

Vamos ahora al formulario de Ventas. El diseño sería idénticoal de compras:



Intentaremos no entretenernos mucho en este formulario, ya que el funcionamiento es similar al de compras. Antes de codificar nada, creamos la correspondiente tabla para Ventas, idéntica a la de compras.



Vamos al código. El Form_Load es el mismo, solo que abriremos la tabla de ventas en vez de la de entradas. El resto, sería igual:

```
Private Sub Form_Load()
   Set Rs = New ADODB.Recordset
   Rs.Open "ventas", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
   Lista.ListItems.Clear
   If Not Rs.EOF Then
      While Not Rs.EOF
         With Lista.ListItems.Add(, , Rs("Id"))
            .SubItems(1) = Rs("Nombre")
            .SubItems(2) = Rs("Sabor")
            .SubItems(3) = Rs("Cantidad")
            .SubItems(4) = Rs("Fecha")
         End With
         Rs.MoveNext
      Wend
      MoverDatosACampos
   Else
      MsgBox "La tabla de ventas está vacía", vbExclamation, "Aviso"
      CmdModificar.Enabled = False
      CmdBorrar.Enabled = False
   End If
   Deshabilitar
End Sub
```

Las rutinas para mover los datos a campos, deshabilitar, deshabilitar botones, etc.. son las mismas, ya que vamos a llamar a los controles (botones, lista, TextBox) con los mismos nombres que en el formulario de entradas. Otra de las rutinas que debemos modificar es el botón Validar, ya que ahora no tenemos que sumar productos, sino restarlos, comprobando antes si en el stock hay suficientes helados como para realizar la venta. En caso de que el producto no exista o supere la cantidad del stock, mostraremos un mensaje indicativo de ello. Estos serían los cambios del procedimiento Validar:

```
Private Sub CmdValidar_Click()
   For i = 1 To Lista.ListItems.Count
      If Lista.ListItems(i).Checked Then
         Rs.MoveFirst
         Rs.Find "Id=" & Lista.ListItems(i).Text
         If Not Rs.EOF Then
            Set RsAlta = New ADODB.Recordset
               RsAlta.Open "Select * from stock where Nombre='" &
                  Rs("Nombre") & "' and Sabor='" & Rs("Sabor")
                  & "'", Conn.ConnectionString, adOpenDynamic,
                  adLockOptimistic
               If Not RsAlta.EOF Then
                  If Rs("Cantidad") <= RsAlta("Cantidad") Then
                     RsAlta("Cantidad") = RsAlta("Cantidad") - Rs("Cantidad")
                     RsAlta.Update
                  Else
                     MsgBox "El stock es inferior a la cantidad solicitada", vbInformation, "Aviso"
                     Exit Sub
                  End If
               Else
                  MsgBox "Este producto no existe en el Stock", vbExclamation, "Aviso"
                  Exit Sub
               End If
               RsAlta.Close
               Rs.Delete
            End If
         End If
   Next
   Lista.ListItems.Clear
   Call Form_Load
End Sub
```

Pasamos a comentarlo:

Principalmente hemos creado dos condiciones más. La primera, si no se encuentra el registro en la tabla de stock, mostramos un mensaje indicándolo. En el caso que si se encuentre el registro, pero sin embargo, la cantidad solicitada sea superior a la del stock, mostramos otro mensaje:

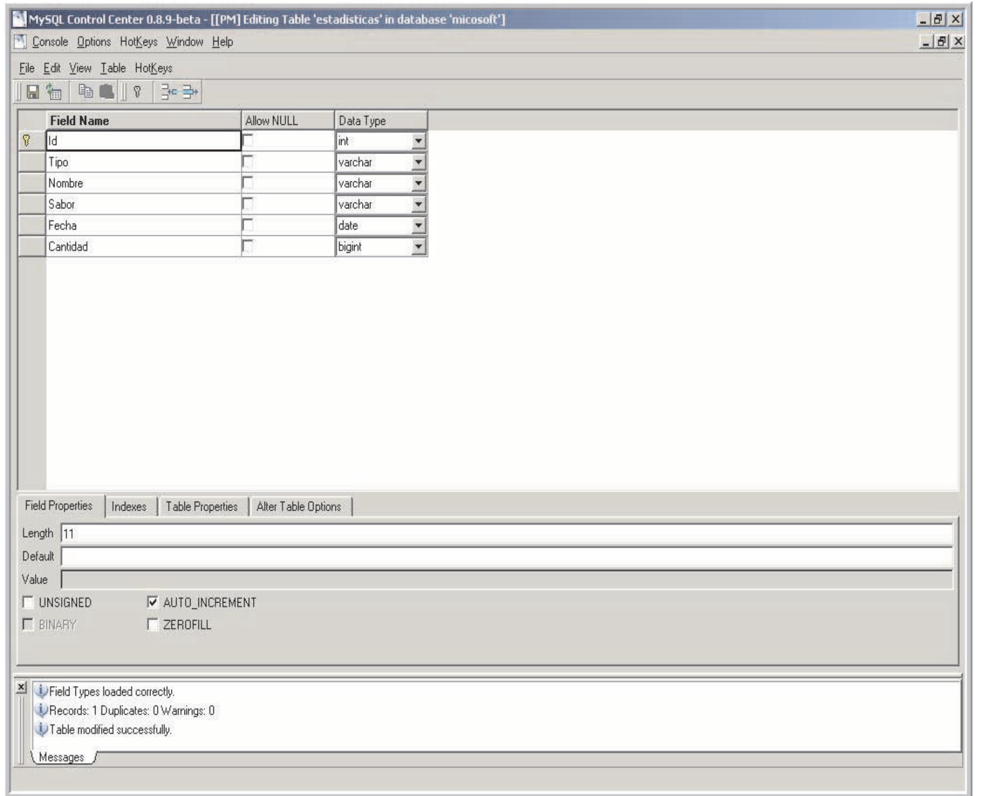
```
If Rs("Cantidad") <= RsAlta("Cantidad") Then
```

Bueno, con este formulario podemos dar por cerrado el mantenimiento de productos.

Si ahora os digo que este proyecto no cumple ninguna de las reglas de normalización de base de datos, supongo que no os sentaría bastante bien, pero es así. Y lo hemos hecho así porque quiero que seáis vosotros los que lo arregléis, siguiendo las premisas que explicaré al acabar el formulario de Estadísticas.

Vamos a crear una nueva tabla que gestione estas ventas y entradas. Para entendernos, será un histórico de estas mismas.

Esta sería la estructura:



Vamos a crear un proceso tanto en el formulario de entradas como en el de ventas para que, cada vez que se valide algún producto, este cree un registro en esta tabla, indicando en el campo Tipo si es una venta o entrada.

En este caso utilizaremos como ejemplo el formulario de Entradas, teniendo en cuenta que el de Ventas tendrá el mismo código, con la única diferencia que al campo Tipo le enviaremos un valor u otro. Para empezar, abriremos el Recordset en el Form_Load:

```
Private Sub Form_Load()
    Set Rs = New ADODB.Recordset
    Set RsEstadisticas = New ADODB.Recordset
    Rs.Open "entradas", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
    RsEstadisticas.Open "Estadisticas", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
………
………
```

Y para dar de alta en la tabla de estadísticas, una sencilla rutina:

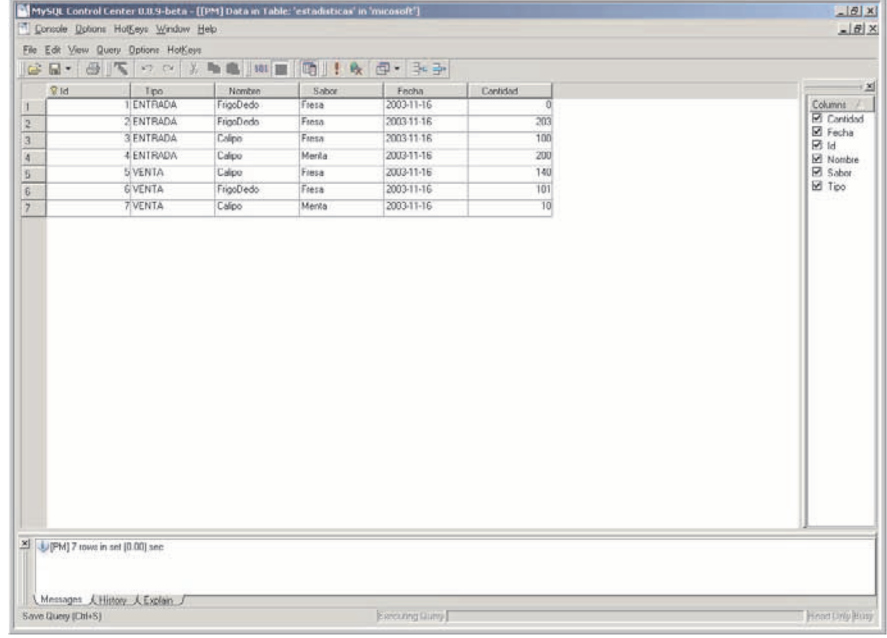
```
Sub AnadirEstadistica()
    RsEstadisticas.AddNew
    RsEstadisticas("Tipo") = "ENTRADA"
    RsEstadisticas("Nombre") = Rs("Nombre")
    RsEstadisticas("Sabor") = Rs("Sabor")
    RsEstadisticas("Fecha") = Rs("Fecha")
    RsEstadisticas("Cantidad") = Rs("Cantidad")
    RsEstadisticas.Update
End Sub
```

Que la llamaremos justo antes de borrar el registro en la correspondiente tabla:

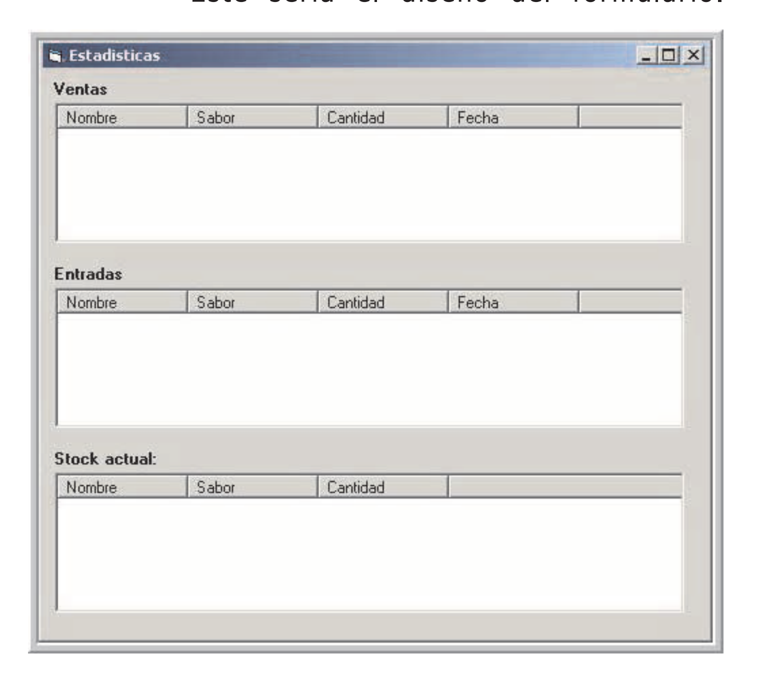
```
        End If
        RsAlta.Close
        AnadirEstadistica
        Rs.Delete
```

Para el formulario de ventas, haremos exactamente lo mismo, con la única diferencia que al campo Tipo moveremos el literal "VENTA", en lugar de "ENTRADA"

Para comprobar que funciona, podemos hacer algunas ventas y entradas, mirando posteriormente el contenido de la tabla desde el gestor de base de datos.



Ahora vamos al formulario de estadísticas. El diseño se comprende en tres listas, una para controlar las ventas por fecha, otra para las entradas, y una última para el stock actual. Este sería el diseño del formulario.



Lo interesante del formulario de estadísticas es que podemos sumariar (obtener una lista) las cantidades vendidas un mismo día, así como las entradas, de un mismo producto. Empecemos a picar el código…

Estas son las variables a utilizar

```
Dim RsVentas As ADODB.Recordset
Dim RsEntradas As ADODB.Recordset
Dim RsStock As ADODB.Recordset
```

En el evento Load del formulario las instanciaremos

```
Private Sub Form_Load()
    Set RsVentas = New ADODB.Recordset
    Set RsEntradas = New ADODB.Recordset
    Set RsStock = New ADODB.Recordset
```

Y empezamos a abrir los Recordset, pero esta vez utilizando una sentencia SQL algo más complicada que las demás.

```
    RsVentas.Open "Select Tipo, Nombre, Sabor, Sum(Cantidad)
as Cantidad, Fecha from estadisticas  where Tipo='VENTA' group
by Fecha, Nombre, Sabor Order By Fecha Desc",
Conn.ConnectionString, adOpenDynamic, adLockOptimistic

    ListaVentas.ListItems.Clear
    ListaEntradas.ListItems.Clear
    ListaStock.ListItems.Clear
```

Voy a explicar la "Query". En la cláusula "Select" indicamos los campos que queremos seleccionar. Estos van a ser el Tipo, Nombre, Sabor, Fecha y el sumatorio de las cantidades, al cual llamaremos Cantidad.

La cláusula "From" es sencilla, tan solo incluimos el nombre de la tabla donde se encuentran los campos descritos anteriormente.

Vemos que hemos incluido una nueva cláusula, el "Group By". Como su traducción indica (Agrupa Por), aquí decimos por qué campos queremos agrupar los registros.

En nuestro caso, agruparemos por la Fecha, el Nombre y Sabor.

La cláusula "Where" filtra los registros para que solo aparezcan aquellos que sean ventas, y el "Order By" ordena por el campo que queramos.

El resultado final es que se sumarán las cantidades de aquellos registros que tengan en común la Fecha, Nombre y sabor.

Ahora cargamos la lista:

```
     While Not RsVentas.EOF
         With ListaVentas.ListItems.Add(, , RsVentas("Nombre"))
             .SubItems(1) = RsVentas("Sabor")
             .SubItems(2) = RsVentas("Cantidad")
             .SubItems(3) = RsVentas("Fecha")
         End With
         RsVentas.MoveNext
    Wend
    RsVentas.Close
```

No nos olvidemos cerrar también el Recordset, ya que

consumimos memoria inútilmente si no lo hacemos. Posteriormente realizamos la misma acción contra la otra tabla, la de entradas.

```
RsEntradas.Open "Select Tipo, Nombre, Sabor, Sum(Cantidad) as Cantidad, Fecha from estadisticas  where Tipo='ENTRADA' group by Fecha, Nombre, Sabor Order By Fecha Desc", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
While Not RsEntradas.EOF
    With ListaEntradas.ListItems.Add(, , RsEntradas("Nombre"))
        .SubItems(1) = RsEntradas("Sabor")
        .SubItems(2) = RsEntradas("Cantidad")
        .SubItems(3) = RsEntradas("Fecha")
    End With
    RsEntradas.MoveNext
Wend
RsEntradas.Close
```

Vemos que la única variación de la SQL está en la cláusula “Where”, donde en lugar de buscar las ventas, buscamos las entradas, cambiando el valor de búsqueda (where Tipo='ENTRADA').

Para la lista del Stock no tenemos complicaciones, ya que mostraremos el contenido de la tabla tal cual

```
RsStock.Open "Stock", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
While Not RsStock.EOF
    With ListaStock.ListItems.Add(, , RsStock("Nombre"))
        .SubItems(1) = RsStock("Sabor")
        .SubItems(2) = RsStock("Cantidad")
        '.SubItems(3) = RsStock("Fecha")
    End With
    RsStock.MoveNext
Wend
RsStock.Close
```

Yo no creo necesario que en este formulario se deba poder realizar cualquier otra acción, así que lo dejo a vuestra elección.

Para obtener un buen resultado en la ventana de estadísticas, os recomiendo que deis de alta entradas o ventas con diferentes fechas, así se detallarán mejor:

Estadisticas

Ventas

Nombre	Sabor	Cantidad	Fecha
Calipo	Fresa	40	16/11/2004
Calipo	Platano	100	16/11/2004
Calipo	Fresa	150	16/11/2003
Calipo	Menta	10	16/11/2003
FrigoDedo	Fresa	101	16/11/2003

Entradas

Nombre	Sabor	Cantidad	Fecha
Calipo	Fresa	100	16/11/2003
Calipo	Menta	200	16/11/2003
FrigoDedo	Fresa	203	16/11/2003

Stock actual:

Nombre	Sabor	Cantidad
FrigoDedo	Fresa	302
Calipo	Menta	190
Calipo	Platano	111
Calipo	Fresa	609
FrigoPie		0
Pepe	as	1

Bien, ya tenemos el formulario de estadísticas, y como lo prometido es deuda, paso a explicaros lo que antes os había comentado.

Me gustaría que entendieseis que, cuando estamos dando de alta productos, ventas o entradas, estamos introduciendo los helados, como diría..., “a pelo”. Esto nunca se hace así. Lo normal sería tener una tabla llamada “helados” o “productos” donde tendríamos todos nuestros productos. Por ejemplo, esta tabla podría tener los campos Id, Nombre, Sabor. Los campos Nombre y Sabor no deben repetirse, ya que sería absurdo tener dos productos iguales. Una vez tengamos esta tabla, sería interesante crear un formulario para el mantenimiento de estos productos.

El formulario no sería muy complicado. Desde él daremos de alta, modificaremos y eliminaremos helados.

Una vez acabado esto, deberíamos cambiar los formularios de stock, entradas y ventas, en la parte que comprende a los helados. Ya no deberíamos permitir que los helados se introduzcan en cajas de texto, sino que deberíamos tener unos combos desplegables con el nombre, así como el sabor en otro combo.

El campo de cantidad seguiría siendo igual. Esto debemos aplicarlo a los 3 formularios, y así normalizaríamos las 3 tablas. Claro está que también tendremos que cambiar el código del formulario de estadísticas.

Espero que esta explicación os haya servido para plantearos otra forma de ver un mismo proyecto.

Y con esto quiero despedirme definitivamente, ya que doy por acabado el curso de Visual Basic. Espero haberos hecho despertar el gusanillo de la programación, y que ahora mismo muchos de vosotros estéis pensando en desarrollar proyectos por vuestra cuenta. Seguramente en un futuro cercano volveré con otros artículos, pero por ahora necesito tomarme un descanso, para reorganizar mis ideas, y poder plasmarlas.

Sin más, os dejo, encantado de que a mucha gente, este curso, le haya servido para emprender o reemprender su curiosidad hacia el mundo de la programación. Un Saludo, Pedro del Valle

Importante: Para cuando leas estas páginas habremos estrenado una nueva WEB. Allí podrás encontrar el código completo tanto del curso de Visual Basic como cualquier otro editado en la revista, muchos artículos liberados y muchas más cosas que no se han podido hacer antes debido a la poca operatividad de la antigua Web. Gracias por vuestra paciencia.

PROGRAMACION EN GNU LINUX

DESARROLLO DE APLICACIONES EN ENTORNOS UNIX E INICIACIAON AL LENGUAJE C (IV)

el_chaman. LUIS U. RODRIGUEZ PANIAGUA

- Seguimos con nuestro particular curso de programación en LINUX.
- Crearemos Librerías
- Examinaremos la Archivos de Cabecera
- Empezaremos a jugar con el Makefile

1. Presentación

En el último artículo se presentó el código de un programa ya bastante extenso en C. Hoy trabajaremos con dicho código para explicar determinados conceptos con los que ya terminaremos la "parte teórica de C" y que a partir del siguiente número nos permitirá centrarnos más en la programación del sistema. Los conceptos que abordaremos en el presente artículo son:

• Ejemplo de uso de las funciones explicadas en el anterior artículo.

• Creación de nuestras propias librerías de funciones mediante archivos de cabecera.

• Inclusión de macros en nuestro código.

• Creación de archivos Makefile

Como a partir de ahora gran parte del trabajo va a estar a vuestro lado de los monitores pido dos cosas:

• Intentad haced todos los ejercicios y trabajos de investigación que se os sugieran en los artículos. Como nadie nace aprendido estoy a vuestra contínua disposición en el foro de la revista o en su defecto en la dirección de correo luisb@es.gnu.org. También podéis realizar vuestras consultas o ver las soluciones de los ejercicios propuestos a través de la dirección http://leonov.servebeer.com:2525/~luis/ej_articulos.php, pero advierto que el lugar central de consulta será el foro. La razón de ello es que, aunque fuese deseable, no tengo el tiempo ni los medios para realizar un seguimiento personalizado de los ejercicios de cada lector. Espero que de esta manera todos avancemos más rápido.

• No existen preguntas estúpidas. Así que por muy claro que os parezca que esté explicado algo, no dudéis en pedir mayores explicaciones. Os aseguro que por mi parte no será ninguna molestia, así que ¡¡¡ HACED PREGUNTAS !!!. Es más que probable que a veces se me vaya la mano y cometa algún error de bulto. Afortunadamente no soy perfecto :o).

2. Archivos de Cabecera y Librerías

Como ya hemos comentado en alguna ocasión, los archivos de cabecera van a hacer posible que los archivos de código fuente que componen nuestros programas cumplan con la modularidad. Es decir, agruparemos funciones, tipos de datos, etcétera que tengan algo en común en archivos concretos.

Otra ventaja a la hora de hacer esto, es que el código será fácilmente reutilizable. Es decir, a

la hora de hacer nuevos programas que utilicen funciones que hicimos previamente para otros programas, bastará con copiar (o no) los archivos adecuados a la nueva posición.

A continuación vamos a ver como se realiza esto para examinar a continuación, el código fuente del mes pasado. El ejemplo que veremos es muy, muy sencillo en cuanto a código, dado que lo que queremos es centrarnos en la creación de una librería.

2.1. Creando archivos de cabecera

Imaginemos que vamos a hacer una simple aplicación que requiera del uso de operaciones matemáticas. Como es posible que en el futuro necesitemos alguna de estas funciones (es un suponer), decidimos meterlas en archivos totalmente independientes. El primer archivo que crearemos será mates.h y tendrá el siguiente aspecto:

```
        /* Archivo mates.h */
#ifndef _MATES_H_                                   ❶
#define _MATES_H_

/* Definición de constantes */                      ❸
#define PI 3.141593
#define NE 2.718281828

/* Declaración de funciones */                      ❹

/*
* Función: sumar
* Recibe:
*       a: Valor real. Sumando. (parámetro)
*       b: Valor real. Sumando. (parámetro)
* Devuelve:
*       suma de a + b: Valor real
*/
float sumar(float a, float b);

/*
* Función: restar
* Recibe:
*       a: Valor real. Restando. (parámetro)
*       b: Valor real. Restando. (paaámetro)
* Devuelve:
*       resta de a - b: Valor real
*/
float restar(float a, float b);

/*
* Función: multiplicar
* Recibe:
*       a: Valor real. Multiplicando. (parámetro)
*       b: Valor real. Multiplicando. (parámetro)
* Devuelve:
*       producto de a * b: Valor real
*/
float multiplicar(float a, float b);

/*
* Función: dividir
* Recibe:
*       a: Valor real. Dividendo. (parámetro)
*       b: Valor real. Divisor.   (parámetro)
*       e: Valor entero. Código de error (referencia)
* Devuelve:
*       división de a / b: Valor real
*  Si b es 0, se deposita en e el valor -1, invalidando
*   de este modo el resultado de la división. En otro
*  caso el valor de e será 0, indicando que el valor
*  devuelto es correcto.
*/
float dividir(float a, float b, int *e);

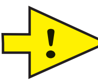
/*
* Función: cambia_signo
* Recibe:
*       a: Valor real. (parámetro)
* Devuelve:
*       Nada
*  Cambiamos el signo del número a.
*/
void cambia_signo(float *a);                        ❺

#endif                                              ❷
```

❶ Comienza el archivo de cabecera. Aquí nos preguntamos si el archivo ha sido previamente cargado. La manera de hacerlo es preguntarle al preprocesador si no se ha definido previamente la constante _MATES_H_. Si no ha sido definida, el procesador incluirá todo lo que hay por debajo hasta el #endif

❷ Aquí marcamos la parte final del archivo de cabecera.

❸ Dentro de un archivo de cabecera pueden ir declaraciones de constantes que previamente podrán ser usadas (o no) por el programa que haga uso de este archivo.



Para aquellos...

Para aquellos que odian las matemáticas… os recordamos que la constante PI se refiere al número matemático llamado "pi" y que su valor es una constante que vale 3.141593, y la constante NE se refiere al número matemático "e" y que su valor es una constante que vale 2.718281828. Estas constantes se utilizan mucho en ecuaciones trigonométricas y cálculo de límites.

❹ El mayor empleo sin duda será el de declarar funciones. Observemos que en el archivo de cabecera sólo ponemos la declaración de la funciones y no su implementación. O dicho de otra manera, decimos todos los datos relevantes de una función (valor de retorno, nombre y parámetros) pero no "como hace las cosas".

Esta manera de hacer las cosas tiene sus ventajas: Por una parte, de un simple vistazo a un archivo de cabecera, podemos obtener toda la información relevante con respecto a su funcionalidad. Suele ser el sitio idóneo para poner los comentarios relativos a las funciones (recordad la importancia de documentar nuestro código). Dónde se ponga la documentación es algo que depende mucho del gusto personal del programador. Pero juzgad por vosotros mismo si con sólo ver este fichero ya seriáis capaces o no de sacarle provecho. Pronto veremos una situación en que será imperativo hacerlo así.

❺ Observemos que ya, incluso en un programa tan sencillo, vamos a hacer uso de términos vistos en anteriores artículos como paso de parámetros por referencia y, por ende, punteros.

En resumidas cuentas; el archivo de cabecera será la tarjeta de presentación de una biblioteca: Nos dice que tipos, constantes y funciones posee, y con esta información, podemos ya incluirla en nuestros programas y trabajar con ella.

Pero aún falta "algo". Cuando hemos visto funciones, hemos hecho hincapié en como decirle en una función lo que debe y como lo debe hacer. En el archivo visto, por ejemplo, en ningún lado decimos cómo se sumarán los números, o cuando sabremos que la división no se puede hacer.

Retomando conceptos de número anteriores, debemos de empezar a acostumbrarnos a diferenciar entre declarar una función (decir cómo es) y su definición (decir cómo funciona). Lo que hemos hecho en el archivo mates.h ha sido declarar las funciones (y alguna constante) que formarán parte de la biblioteca. ¿Dónde irán las definiciones?

Las definiciones irán en un archivo llamado mates.c cuyo contenido será:

```
/*
//
// ARCHIVO: mates.c
//
// DESCRIPCIÓN:
//  Este archivo corresponde al archivo de
// definiciones de la  cabecera
//  asociada en el que definimos las funciones
// declaradas previamente
//
 // AUTOR: (c) 2003 el_chaman
//  Se permite la copia y la distribución de este
// código, así como
 //  su modificación conforme a los  términos de la
// licencia GPL
//  disponible en www.gnu.org.
//
//////////////////////////////////////////////////////////////// */
#include "mates.h"
/* Definición de funciones */

float sumar(float a, float b)
{
  return a + b;
}

float restar(float a, float b)
{
  return a - b;
}
```

```
float multiplicar(float a, float b)
{
 return a * b;
}

float dividir(float a, float b, int *e)
{
if (b == 0) /* caso de división por 0 */
          {
        *e=-1;
        return 0;
        }
        else
        {
         *e=0;
        return a/b;
        }
}

void cambia_signo(float *a)
{
*a = (-1) * *a;
}
```

De entrada, el archivo es un poco más duro de leer. No hay muchos comentarios explicativos ni tan siquiera para las cosas nuevas que nos encontramos. La razón de ello es que habitualmente cuando nos hacemos una biblioteca pueden suceder varias cosas (siempre suponiendo que la vaya a usar un programador):

• Que el programador seamos nosotros, y pasado un tiempo, sólo queramos usar la librería, confiando en que cuando la hicimos, la hicimos bien, sin comernos mucho la cabeza por cosas que hicimos en el pasado (vamos, como la vida misma).

• Que el programador seamos nosotros, pero que tras meter el código del archivo en una biblioteca binaria (veremos qué es eso más adelante en este mismo artículo), el archivo .c asociado al .h se "pierda" y sólo dispongamos de fuente de información el .h (como hayamos "perdido este, mal lo llevamos).

• Que queramos ponerlo a disposición de otros programadores, pero sin que estos husmeen en cómo hemos hecho el código. Entonces meteremos lo correspondiente al archivo .c en una biblioteca binaria y distribuiremos la biblioteca binaria junto con los archivos .h, pero sin los .c. Seguro que se os ocurren ejemplos muy conocidos.

• Y finalmente que distribuyamos nuestro código fuente de manera íntegra y permitamos que cualquiera pueda moddificar, por ejemplo, cómo se dividen los números.

Bien, volviendo al archivo mates.h, hay una cosa que nos debería de llamar la atención:

```
....
#include "mates.h"
....
```

Veamos dos razones por las que nos debería de llamar la atención:

• La primera, más técnica, está referida a la necesidad de incluir esta línea. No olvidemos que en el archivo mates.h están incluídas las declaraciones de las funciones. Esto implica que cuando declaramos una función, tarde o temprano debemos definirla. Dado que debemos definir todas las funciones declaradas en el archivo de cabecera y utilizar todos los datos de este archivo, debemos de cargarlo. De esta manera cuando este archivo se convierta en binario, quedará asociado con el archivo de cabecera correspondiente mediante la oportuna tabla de símbolos.

• La segunda, más trivial, está referída al empleo de "...." en vez de <....>. Es decir, ¿por qué ponemos *#include "mates.h"*

en vez de *#include <mates.h>*?.

La razón es sencilla: Cuando ponemos <....>, le estamos diciendo al compilador que el archivo de cabecera a incluir está dentro de los directorios en los que por defecto busca el compilador archivos de cabecera.

Por el contrario al poner "....", le estamos diciendo que puede que no está en ese árbol de búsqueda, por lo que le vamos a indicar la ruta. Como en su día vimos, la ruta (path)

puede ser tanto una ruta relativa al directorio actual como absoluta. Por lo tanto podremos encontrarnos códigos que contengan cosas como:

```
#include "red/sniffer.h"
#include "../grafs/iconos.h"
#include "/home/pedro/includes/berberechos.h"
        ...
        ...
```

Al poner sólo #include "mates.h", el compilador asume que el archivo se encuentra en el mismo directorio desde el que es invocado.

Podría parecer tras lo expuesto en esta parte que el archivo de definiciones siempre tiene que llamarse igual que el archivo de cabecera. Nada más lejos de la verdad.

Incluso podemos emplear varios archivos de definición, tal y como muestra el siguiente micro ejemplo:

• Archivo de cabecera cabecera.h

```
#ifndef _CABECERA_H_
#define _CABECERA_H_
void imprime_error(int codigo);
float division(float a, float b);
#endif
```

• Archivo de definicion 1 input_output.c

```
#include "cabecera.h"
void imprime_error(int codigo)
{
   printf("\n Se encontró un error con código %i \n",codigo);
}
```

• Archivo de definición 2 mas_mates.h

```
#include "cabecera.h"
float division(float a, float b)
{
   ....
   /* Código ya visto */
   ....
}
```

2.2. Compilando

El siguiente paso es realizar un programa que haga uso de nuestra librería. Para ello emplearemos el siguiente programa de ejemplo:

```
/*
//
// ARCHIVO:  prueba00.c
//
// DESCRIPCIÓN:
//   Primer ejemplo que muestra el empleo de
// archivos de cabecera.
//   Compilar con:
//
//         gcc prueba00.c mates.c -I./ -o prueba00
//
// AUTOR: (c) 2003 el_chaman
// Se permite la copia y la distribución de este código,
// así como
// su modificación conforme a los  términos de la
// licencia GPL
// disponible en www.gnu.org.
//
/////////////////////////////////////////////////////////////////// */
#include <stdio.h>

#include "mates.h"
                                                              ❶
main()
{
      /* Declaración de variables */
      float numA, numB, resultado_parcial;
      int flag_error;

      /* Lectura de datos por teclado */
      printf("\n Deme un número real (numA): ");
      scanf("%f",&numA);
      printf("\n Deme un número real (numB): ");
      scanf("%f",&numB);

      /* Realizamos operaciones */
      resultado_parcial = sumar(numA, numB);
      printf("\n %f + %f = %f ", numA, numB, resultado_parcial);

      resultado_parcial = restar(numA, numB);
      printf("\n %f - %f = %f ", numA, numB, resultado_parcial);
```

```
        resultado_parcial = multiplicar(numA, numB);
        printf("\n %f * %f = %f ", numA, numB, resultado_parcial);

        resultado_parcial = dividir(numA, numB, &flag_error);
        if( flag_error == -1)
        {
                printf("\n ERROR: División por 0");
        }
        else
        {
                printf("\n %f / %f = %f ", numA, numB, resultado_parcial);
        }
        cambia_signo(&numA);
        cambia_signo(&numB);
        printf("\n Los valores de numA y numB tras cambiarlos de \
        signo, son respectivamente %f y %f\n", numA, numB);
}
```

❶ Obsérvese que

• Cargamos el archivo de cabecera en el que se encuentran declaradas las funciones que vamos a usar.

• Seguimos empleando "...."

Bien, ya lo tenemos todo preparado para compilar nuestro primer programa compuesto por múltiples archivos. Primero vamos a compilarlo de manera que sea el compilador quien trabaje por nosotros.

Posteriormente, haremos una compilación detallada para saber cuál es el trabajo del que exactamente nos ha librado el compilador.

A continuación, realizaremos los pasos necesarios para crear una librería binaria y, finalmente, crearemos un archivo Makefile para automatizar todos los paso necesarios.

Por de pronto podéis bajaros el código fuente correspondiente a estos archivos de http://leonov.serveftp.com:2525/~luis/descargardor.php?descarga=descargas/a8_ej00.tgz&nombre=a8_ej00 y descomprimirlos.

2.2.1. Primera aproximación a la compilación.

La manera de compilar este programa en una medida estándar de tiempo plís-plás es mediante la siguiente instrucción:

gcc mates.c prueba00.c -o prueba00

Antes de hacer nada más, vamos a hacer un experimento por si alguien se despistó en apartados anteriores.

Abrid con vuestro editor de texto el archivo prueba00.c.

Cambiad la línea donde pone "mates.h" por <mates.h> (¿pero no habíamos quedado en que....? Ya le dio el siroco).

Ahora volved a compilar. Os debe de salir algo como esto:

gcc mates.c prueba00.c -o prueba00

prueba00.c:18:19: mates.h: No existe el fichero o el directorio

Vale, Yogurtu Ngué, has triunfado... Como si no viese errores de compilación todos los días...

Ahora, **sin tocar nada del código**, compilad con la siguiente instrucción:

gcc mates.c prueba00.c -I./ -o prueba00

Primer ejercicio: Según lo que se ha explicado hoy aquí, y con la ayuda de man gcc, ¿quién puede explicar lo que ha sucedido en la tercera compilación? ¿Y en la primera? ¿Y en la segunda?

Volviendo a lo que nos ocupa. En la compilación inicial decimos al compilador que genere los archivos objetos necesario de manera temporal (el compilador los generará probablemente en /tmp), los enlace y genere el ejecutable, llamado prueba00.

2.2.2. Segunda aproximación a la compilación: Creación de librerías

Antes de continuar, volvemos a sustituir en prueba00.c, <mates.h> por "mates.h"

Vamos a aplicar ahora lo aprendido en días pasados.

En primer lugar vamos a crear el código objeto correspondiente a mates.c; luego el correspondiente a prueba00.c y finalmente los enlazaremos:

```
$ gcc -c mates.c ❶
$ gcc -c prueba00.c ❷
$ gcc mates.o prueba00.o -o prueba00❸
```

❶ Generamos el archivo objeto binario para mates.c

❷ Generamos el archivo objeto binario para prueba00.c

❸ Enlazamos los archivos objeto binario y creamos el ejecutable.

Aunque a primera vista pudiera parecer que es un proceso largo y tedioso, veremos que es muy importante. Pero como siempre vamos a ponernos en una situación en la que todo parece ir mal, para explicar que sucede.

Antes de nada vamos a borrar los archivos objeto binario: *rm *.o*.

Ahora ejecutemos:

```
$ gcc prueba00.c -o prueba00
/tmp/cc6mmzeX.o(.text+0x5c): En la función `main':
: undefined reference to `sumar'
/tmp/cc6mmzeX.o(.text+0x92): En la función `main':
: undefined reference to `restar'
/tmp/cc6mmzeX.o(.text+0xc8): En la función `main':
: undefined reference to `multiplicar'
/tmp/cc6mmzeX.o(.text+0x105): En la función `main':
: undefined reference to `dividir'
/tmp/cc6mmzeX.o(.text+0x148): En la función `main':
: undefined reference to `cambia_signo'
/tmp/cc6mmzeX.o(.text+0x153): En la función `main':
: undefined reference to `cambia_signo'
collect2: ld devolvió el estado de salida 1
```

Eso: ¡jaarrrlll!

Ahora tecleemos:

```
$ gcc -c mates.c
gcc mates.o prueba00.c -o prueba00
```

Como vemos, teniendo la información binaria de las funciones, el compilador sabe como crear un ejecutable. Aún no me he terminado de ir por las ramas. Creemos estos archivos:

```
/*
    //
    // ARCHIVO: error.c
    //
    // DESCRIPCIÓN:
    //
    //
    // AUTOR: (c) 2003 el_chaman
    // Se permite la copia y la distribución de este
    // código, así como
    // su modificación conforme a los  términos
    // de la licencia GPL
    // disponible en www.gnu.org.
    //
    //////////////////////////////////////////////////////// */
    #include "error.h"
    void imprime_error(int cod)
    {
     printf("\n Código de error devuelto: %i \n", cod);
    }
```

Y

```
/*
    //
    // ARCHIVO: error.h
    //
    // DESCRIPCIÓN:
    //
    //
    // AUTOR: (c) 2003 el_chaman
    // Se permite la copia y la distribución de este
    // código, así como
    // su modificación conforme a los  términos
    // de la licencia GPL
    // disponible en www.gnu.org.
    //
    //////////////////////////////////////////////////////// */
    #ifndef _ERROR_H_
    #define _ERROR_H_
    void imprime_error(int cod);
    #endif
```

Además vamos a añadir las siguientes líneas al archivo prueba00.c (pongo las líneas anteriores como referencia):

Personalmente voy a copiar prueba00.c a prueba01.c para poder distribuir los dos archivos (sin y con modificación) en el paquete de descarga. Los siguiente ejemplos asumen que los cambios se realizan en prueba00.c

```
#include <stdio.h>
#include "mates.h"
#include "error.h" /* línea nueva */
....
....
        printf("\n ERROR: División por 0");
        imprime_error(flag_error); /* línea nueva */
        ....
        ....
```

Bien, la familia crece. Ahora siguiendo anteriores pasos:

```
$ rm *.o prueba00
$ gcc -c error.c
$ gcc -c mates.c
$ gcc error.o mates.o prueba00.c -o prueba00
```

Por lo visto anteriormente sabemos que en los archivos .o están almacenadas las funciones en formato binario, de manera que tras enlazarlas, pueden pasar a formar parte de un ejecutable.

¿Y si pudiésemos empaquetar todas esas funciones de manera que al distribuir ese paquete ya no fuesen necesarios los archivos .c? ¿Y si tan ni siquiera fuese necesario distribuir dicho paquete porque sabemos, por ejemplo, que todos aquellos a los que vamos a enviar dicho paquete ya poseen una versión del mismo?

Pues bien, este misterioso paquete es la librería. Una librería, informalmente no será más que eso: un paquete de archivos binarios que contienen funciones.

Para construir un paquete teclearemos:

```
$ ar r libmio.a error.o
$ ar r libmio.a mates.o
```

Literalmente estamos diciendo: Introduce en la librería libmio.a, los archivos error.o y mates.o.

Ahora, para compilar bastaría con teclear:

```
$ gcc gcc  prueba00.c libmio.a -o prueba00
```

El comando ar (ver man ar) será el constructor de bibliotecas.

Otro comando muy útil asociado con bibliotecas será nm, que nos permite examinar el contenido de estas.

```
$ nm libmio.a

  mates.o:
  00000060 T cambia_signo
  00000021 T dividir
  00000016 T multiplicar
  0000000b T restar
  00000000 T sumar

  error.o:
  00000000 T imprime_error
           U printf
```

De esta manera, a medida que vamos desarrollando nuestros proyectos, podemos ir creándonos librerías en las que tengamos código que solemos emplear muchas veces.

El tiempo que se gana de esta manera en el desarrollo de una aplicación puede ser considerable si seguimos un modelo de programación modular estricto.

Llegados a este punto, y teniendo en cuenta lo visto en los primeros artículos de esta sería, si echamos un vistazo, a lo que hemos hecho obtendremos la siguiente estructura prescindiendo de archivos intermedios:

```
a8_ej00
|-- error.c
|-- error.h
|-- mates.c
|-- mates.h
`-- prueba00.c
```

Lo cual resulta un poco desalentador al recordar la insistencia en colocar cada cosa en su sitio. Tras un poco de bricolaje con los ficheros, el anterior esquema debe de convertirse en:

```
a8_ej00/
|-- bin
|-- doc
|-- include
|   |-- error.h
|   `-- mates.h
|-- lib
`-- src
    |-- error.c
    |-- mates.c
    `-- prueba00.c
```

Vale, muy bonito. Pero ahora ¿cómo demonios realizo los pasos de antes?Pues suponiendo que estamos situados dentro d el directorio a8_ej00, la secuencia de órdenes a ejecutar es:

```
$ gcc -c src/error.c -I./include
$ gcc -c src/mates.c -I./include
$ ar -r lib/libmio.a mates.o
$ ar -r lib/libmio.a error.o
$ gcc src/prueba00.c -lmio -L./lib -I./include -o bin/prueba00
```

Que dejará al directorio en esta situación:

```
a8_ej00/
|-- bin
|   `-- prueba00
|-- doc
|-- error.o
|-- include
|   |-- error.h
|   `-- mates.h
|-- lib
|   `-- libmio.a
|-- mates.o
`-- src
    |-- error.c
    |-- mates.c
    `-- prueba00.c
```

Bien, antes de que nadie abandone este cursillo por otro avanzado de pesca de berberechos, vamos a explicar de donde sale tamaño galimatías.

$ gcc -c src/error.c -I./include

Aquí decimos al compilador: Genérame el código objeto (-c) del archivo error.c que se encuentra en el subdirectorio src y busca las librerías necesarias en el subdirectorio include (-I./include)

$ gcc -c src/mates.c -I./include

Análogo al anterior

$ ar -r lib/libmio.a mates.o

Lo cual interpretamos como introduce el objeto binario mates.o que está en este mismo directorio en la librería libmio.a que está dentro del subdirectorio lib

$ ar -r lib/libmio.a error.o

Análogo al anterior.

Y finalmente:

gcc src/prueba00.c -lmio -L./lib -I./include -o bin/prueba00

Que interpretaremos como Compila el archivo prueba00.c que se encuentra dentro del directorio src y enlázalo con la librería libmio.a (-lmio) que la debes de buscar en el subdirectorio lib (-L./lib) utilizando para los archivos de cabecera el subdirectorio include (-I./include) y depositando el ejecutable resultante en el subdirectorio bin (-o bin/prueba00).

Sí.

Aterrador. Y como supongo que ya sospecháis, existe una manera de convertir todo este proceso en algo mucho más sencillo. Sólo imaginaos que si este pequeño proyecto requiere de comandos "tan" enrevesados (y eso que no hemos puesto ningún error a posta), imaginaos lo difícil que puede resultar ir haciendo un gran proyecto en el que tenemos que compilar, depurar, modificar, compilar... Seguro que fácilmente nos saltamos algún paso.

Resalto de nuevo un detalle: -lmio carga la librería libmio.a. Podemos deducir entonces que todas las librerías que hagamos deben de

comenzar por lib y poseer la extensión .a. Intentad ver que sucede si no se sigue esta norma.

3. Creando un Makefile

3.1. ¿Por qué es necesario usar make?

Pues tras el susto anterior, he aquí la solución: el programa make. Este programa leerá un archivo de texto escrito por nosotros (Makefile y se encargará de llamar a los distintos programas del proceso de compilación en el orden adecuado.

De esta manera, tras escribir el archivo Makefile una única vez, estaremos seguros de que make llama a los programas adecuados en los momentos adecuados, por lo que nos podremos centrar más en nuestra tarea: Introducir código.

Llegados a este punto alguien podría pensar: ¿Qué necesidad tengo de aprender el funcionamiento de un nuevo programa [make] cuándo puedo meter la secuencia de llamadas al compilador en un archivo de script ?

O dicho de otro modo: ¿No podríamos hacer un script que contuviera

```
#!/bin/sh
    #
    # Nombre: compilar.sh
    #
    # Descripción: script que compila mi proyecto
    #
    # Cambiar los permisos a 755 y ejecutar con:
    #
    #       sh compilar.sh
    #
    ########################################
    gcc -c src/error.c -I./include
    gcc -c src/mates.c -I./include
    ar -r lib/libmio.a mates.o
    ar -r lib/libmio.a error.o
    gcc src/prueba00.c -lmio -L./lib -I./include -o bin/prueba00
```

y ejecutarlo?

Por poder, podríamos, pero tendríamos algunas desventajas:

• En primer lugar este archivo no proporciona ningún tipo de información sobre las dependencias que existen entre los ficheros que forman nuestro proyecto. No sabemos qué archivos .c están asociados a archivos .h, por ejemplo. Si tenemos en cuenta que, como hemos visto, varios archivos .c pueden depender de un archivo .h, y que no tienen que llamarse de la misma manera, un segundo programador recebiría muy poca información sobre la estructura de nuestro proyecto.

Si este proyecto crece en número de componentes, la tarea de diseñar y programar un script que lo compile, se convierte por si misma en una tarea de programación comparable al código que estamos realizando y sujeto a los peligros al que estamos sometido en este: fallos en la programación, omisión de instrucciones. Si no tenemos la certeza absoluta de que este script está perfecto (cosa más difícil a medida que el tamaño del proyecto crece), nunca sabremos si una mala compilación del proyecto se debe a que está mal programado el script que lo genera o el propio código fuente del proyecto.

• En segundo lugar debemos de tener en cuenta que este script compila cada vez que se ejecuta todos y cada uno de los archivos del proyecto, generando los archivos objeto una y otra vez y enlazándolos.

Veamos lo inconveniente de esta situación. Imaginemos que de una anterior compilación tenemos ya la librería libmio.a y el ejecutable prueba00. Sin embargo deseamos hacer cambios en el programa principal (prueba00.c).

Dado que los cambios necesarios se han hecho en dicho archivo, bastaría ejecutar sólo la sentencia:

```
gcc src/prueba00.c -lmio -L./lib -I./include -o bin/prueba00
```

para generar la nueva versión del ejecutable.

Sin embargo, nuestro objetivo a la hora de crear el script fue el de no tener que volver a teclear ninguna sentencia de compilación, por lo que lo ejecutamos. Entonces se vuelven a compilar y generar los siguientes archivos: error.o, mates.o, libmio.a, prueba00. Ahora imaginaos un proyecto que conste de 100 archivos fuente que generan, por ejemplo, 75 archivos objeto que a su vez generan 5 librerías y 7 ejecutables. Imaginad que estáis haciendo cambios continuos en uno de los ficheros por motivos de depuración, por ejemplo. En vez de tardar una unidad de tiempo, estaremos tardando... ¡87!. Y así, una y otra vez.

Vemos pues que, a no tan largo plazo, este mecanismo posee unas carencias que no lo dotan como mecanismo óptimo para la generación de proyectos. Es cuando surge la herramienta make que resultará lo suficientemente "inteligente" para no caer en las dos limitaciones vista que posee el empleo de scripts.

3.2. Funcionamiento de make

Básicamente la sintaxis de make es:

make [-f archivo_makefile] [opción] ...
[variable=valor] ... [objetivo] ...

Donde archivo_makefile será el archivo dónde escribamos las sentencias para el programa make. Este archivo puede ser cualquier fichero de texto con cualquier nombre. Al ejecutar make sin ningún parámetro, éste busca sus archivos de instrucción en archivos llamados makefile y Makefile por ese orden. Si nosotros hemos llamado al archivo de sentencias de una manera distinta a makefile o Makefile, por ejemplo, mates_makefile, entonces estamos obligados a invocar a make con la opción -f

make -f mates_makefile

Con respecto al resto de las opciones, en elsiguiente apartado veremos objetivo, opción relativa ejecutar sólo un subconjunto de sentencias dentro de un archivo Makefile. Las demás opciones, por ahora quedan como ejercicio de consulta de man make debido a la extensión del tema. Aún así, con lo que veremos aquí, podemos realizar cualquier proyecto que use make

3.3. Estructura de un archivo Makefile

Una vez que entendemos el funcionamiento del programa make, nos centramos en la estructura de su archivo de sentencias Makefile

Un archivo Makefile consta de sentencias del tipo:

```
objetivo : dependencias
    <TABULADOR>comando1;\
    <TABULADOR>comando2;\
        .....
    <TABULADOR>comandoG;
```

Donde objetivo será un archivo que queremos construir, dependencias una especificación de los archivos necesarios para poder construir objetivo, y comandoX los pasos necesarios para hacerlo.

¡Ojo!: No se ha puesto <TABULADOR> por capricho: es imperativo. Es algo a tener en cuenta en los siguientes ejemplos.

Por poner un ejemplo, Si queremos construir prueba00.o, ¿ de qué archivo o archivos dependeremos ? Pues del archivo que contiene el código fuente, es decir, prueba00.c. Luego una sentencia Makefile que nos podríamos encontrar es:

```
prueba00.o : src/prueba00.c
    gcc -c -I./include -L./lib src/prueba.c
```

Claro que luego alguien puede tener en sus dependencias a prueba00.o:

```
bin/prueba00 : prueba00.o lib/libmio.a
```

```
gcc prueba.o -lmio -L./lib -I./include -o bin/prueba00
```

Normalmente los pasos a seguir son los de poner la instrucción que genera el ejecutable. Esta tendrá sus dependencias que a su vez tendrán unas reglas de construcción (en el ejemplo visto se ve que el ejecutable bin/prueba00 necesita a prueba00.o, pero este a su vez requiere de src/prueba00.c).

Además de las dependencias, decimos como resolverlas; es decir, especificamos las llamadas necesarias al compilador para generar los objetivos.

Para simplificar las cosas, podemos utilizar variables dentro de un Makefile. Un uso normal suele ser:

```
CC = gcc
CC_FLAGS = -L./lib -I./include

bin/prueba00 : prueba00.o lib/libmio.a
        $(CC) prueba.o -lmio $(CC_FLAGS) -o bin/prueba00
```

Teniendo estos conceptos aclarados, veamos como sería nuestro archivo Makefile

```
#
#
# ARCHIVO: Makefile
#
# DESCRIPCIÓN:
#   Archivo para el programa make que compilará
#automáticamente el proyecto.
#   Para compilar el proyecto teclear:
#
#         $  make
#
# AUTOR: (c) 2003 el_chaman
# Se permite la copia y la distribución de este código, así
# como su modificación conforme a los  términos de la
# licencia GPL disponible en www.gnu.org.
#
#//////////////////////////////////////////////////////////////////////
############ Entorno de trabajo ##############
# Directorios del proyecto
DIR_INCLUDE = ./include
DIR_LIBS = ./lib
DIR_SRC = ./src
DIR_BIN = ./bin
# Programas que usaremos para generar nuestro proyecto
# Generador de librerías
AR = ar
# Opciones para el generador de librerías
AR_FLAGS = rs
# Compilador/enlazador
CC = gcc
# Opciones para el compilador/enlazador
CC_FLAGS = -I$(DIR_INCLUDE) -L$(DIR_LIBS)
#
##############################################
#
############# Generación de código ##@########
#
# Archivos a generar
#
# Librería
LIBRERIA = $(DIR_LIBS)/libmio.a
OBJS_LIB = error.o mates.o
OBJS_LIB_SRC = $(DIR_SRC)/error.c $(DIR_SRC)/mates.c
LIB_ENLACE = -lmio
# Ejecutable
PROGRAMA = $(DIR_BIN)/prueba00
OBJS_PROG = prueba00.o
#
# Generación ejecutable
$(PROGRAMA) : $(OBJS_PROG) $(LIBRERIA)
      $(CC) $(OBJS_PROG) $(CC_FLAGS) $(LIB_ENLACE) -o $(PROGRAMA)
# Generación objetos programa
$(OBJS_PROG):
      $(CC) -c $(DIR_SRC)/prueba00.c $(CC_FLAGS)
#Generación librería
$(LIBRERIA) : $(OBJS_LIB)
      $(AR) $(AR_FLAGS) $(LIBRERIA) $(OBJS_LIB)
# Generacion de objetos para la librería
$(OBJS_LIB) :   $(OBJS_LIB_SRC)
      $(CC) -c $(OBJS_LIB_SRC) $(CC_FLAGS)

#
```

Vale, ahora sí que me he perdido.

Aunque a primera vista parezca ser muy complejo, es más sencillo de lo que aparenta. Cojamos la chicha del código:

```
#
# Generación ejecutable
$(PROGRAMA) : $(OBJS_PROG) $(LIBRERIA)
    $(CC) $(OBJS_PROG) $(CC_FLAGS) $(LIB_ENLACE) -o $(PROGRAMA)
# Generación objetos programa
$(OBJS_PROG):
    $(CC) -c $(DIR_SRC)/prueba00.c $(CC_FLAGS)
#Generación librería
$(LIBRERIA) : $(OBJS_LIB)
    $(AR) $(AR_FLAGS) $(LIBRERIA) $(OBJS_LIB)
# Generacion de objetos para la librería
$(OBJS_LIB) :   $(OBJS_LIB_SRC)
    $(CC) -c $(OBJS_LIB_SRC) $(CC_FLAGS)
#
```

y sustituyamos el valor de las variables:

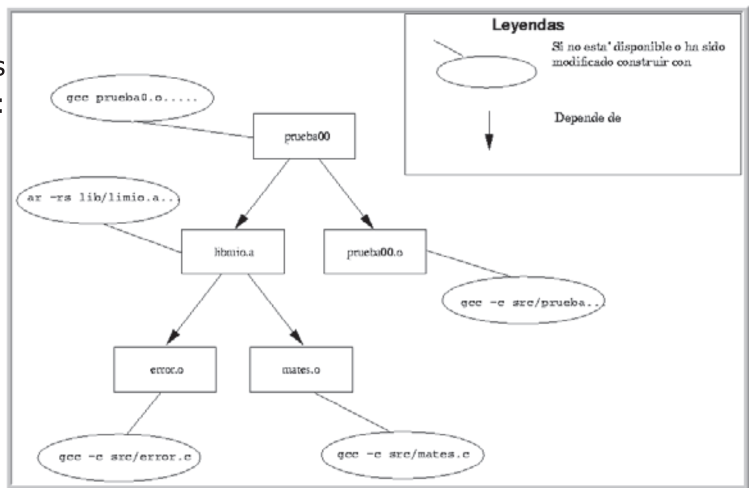
```
#
# Generación ejecutable
bin/prueba00 : prueba00.o  lib/libmio.a
    gcc prueba00.o -I./include -L./lib  -lmio -o bin/prueba00

# Generación objetos programa
prueba00.o:
    gcc -c src/prueba00.c -I./include -L./lib

#Generación librería
lib/libmio.a : error.o mates.o
    ar  rs  lib/libmio.a  error.o mates.o


# Generacion de objetos para la librería
error.o mates.o :  src/error.c src/mates.c
    gcc -c ./src/error.c ./src/mates.c -I./include -L./lib
#
```

Lo revolucionario y lo que nos debe de llamar la atención es el sistema de dependencias. Esto funciona como una estructura de árbol



Cada bloque, para construirse, examina si los bloques de los que depende necesitan ser construidos previamente. Es decir, hasta que los niveles más bajos no están construidos, los más altos no se pueden construir. Si queremos construir un nivel alto, este pasará la pelo a los que dependen de él, y así de manera recursiva hasta que se satisfagan todas las dependencias. Si ponemos un poco de cuidado, construir nuestro propio Makefile no será más que plasmar esta estructura en nuestro archivo.

Ahora es cuando nos deberíamos dar una palmada en la frente y acordarnos de los excelentes artículos de XML que se siguen en esta revista. ¿Algún voluntario para dejar obsoleto al propio Richard Stallman ;o) ? Ya tenéis para qué un ejemplo plas (de los millones que hay) de "para qué me puede servir eso del XML"

Llegados aquí, podemos pensar que realmente no hemos solucionado nada. Es más, todo esto parece mucho más complicado que el scrip, y además hemos escrito mucho más poniendo casi lo mismo que teníamos antes.

Esto será un error por dos razones:

- Ahora, a diferencia que antes, no se compila todo cada vez que invocamos a make, sino sólo aquellas partes que necesitan ser compiladas. El propio programa make se encargará de hacer esto por nosotros.

• Aunque parezca que hemos escrito más, la mayor parte del texto nuevo ha sido para establecer el valor de diversas variables.Por otra parte una vez escrito una archivo Makefile si es necesario realizar cambios en él la modificaciones serán mínimas si este está bien estructurado. En proyectos grandes será algo de agradecer.

Y finalmente, aunque perdamos claridad, el número de instrucciones se puede reducir mucho empleando las macros, los sufijos y otros mecanismos que proporciona make

Finalmente. señalar que podemos construir nuestros propios objetivos con el fin de realizar tareas de mantenimiento. Por ejemplo, si al Makefile de nuestro proyecto le añadimos al final:

```
#
############ Mantenimiento
#
limpia:
  rm *.o lib/* bin/*;
```

Al ejecutar make limpia nos dejará todo el árbol de directorios de nuestro proyecto limpio como al principio.

Intentar hacer la tarea de mantenimiento correspondiente al comando make install

Finalmente, dada la extensión del tema, recomiendo que echéis un vistazo a la documentación de make: http://www.gnu.org/software/make/manual/make.html

Como ejercicio de esta sección, se plantea crear la estructura de subdirectorios necesaria y el Makefile correspondiente para el ejemplo del mes pasado que os lo podéis descargar de: http://leonov.servebeer.com:2525/~luis/descargador.php?descarga=descargas/matriz.tgz&nombre=matriz.tgz

4. Analizando el programa del mes pasado.

4.1. Sobre macros y demás dolores de cabeza

El mes pasado presentamos brevemente un programa que constaba de varios archivos y que pretendía representar el funcionamiento de un tipo de dato nuevo creado por nosotros (Matriz). Tras lo expuesto en el artículo de hoy ya no debe de resultar un misterio el funcionamiento de esos archivos .h y de las extrañas cosas que encontrábamos dentro de ellos.

Aún así sí existen unas cuantas cosas que por falta de espacio no pudieron ser comentadas. Tal vez la que más urgencia corra sea la siguiente instrucción disponible en matriz.h:

```
/* Definimos una macro (código que sustituirá
*POS(pM, i, j) por * (char *) pM->elementos
*+ pM->tamagno * ((i*pM->n) + j) cada vez
* que lo encuentre en el código ) con el fin de
* transformar unas coordenadas del tipo i, j en
 * un desplazamiento sobre una dirección de
 * memoria base referida al puntero
 * pM->elementos

*
*/

#define POS(pM, i, j) ( (char *) pM->elementos + pM->tamagno * ((i*pM->n) + j))
```

Esto es lo que se conoce como una **macro**. Las macros, no son, como pudiera parecer a simple vista, "otra forma de hacer funciones". No. Las macros son porciones de código a los que damos un "mote". Luego, cuando programamos, en vez de poner todo el código, ponemos el "mote" y así nos ahorraremos código.

La sintaxis de una macro será

```
#define mote_macro codigo_a_expandir
```

Pues eso... Funciones.

No. Para empezar las funciones reciben parámetros y devuelven resultados. Las macros NUNCA HACEN ESO. Las macros expanden el "mote" al código al que representan.

Pero si tú has puesto POS(pM, i, j) , como en una función...

¿Seguro? Vamos a utilizar el compilador tal y como aprendimos en su día para salir de dudas:

```
$ cpp matriz.c matriz.i
```

Ahora id al final del archivo matriz.i y observad, por ejemplo la función:

```
void MatrizCambia(Matriz *pM, int i, int j, void *elemento)
{
  memcpy(( (char *) pM->elementos + pM->tamagno *
 ((i*pM->n) + j)), elemento,pM->tamagno);
}
```

Recordemos que esta función, originalmente se declaró en matriz.c como:

```
void MatrizCambia(Matriz *pM, int i, int j, void *elemento)
{
  memcpy(POS(pM,i,j), elemento,pM->tamagno);
}
```

Luego observamos que el procesador ha sustituido (POS(pM,i,j) por ((char *) pM->elementos + pM->tamagno * ((i*pM->n) + j)). Así, sin cambiar valores ni nada; simplemente una sustitución de cadenas.

Pues bien, eso es una macro y como hemos visto es el preprocesador quien tratará con ellas, no el compilador

FE DE ERRATAS --- IMPORTANTE ---

En el número anterior de Hack x Crack, en el artículo de PHP había un error que se repetía en el código. El código PHP empieza por <? Y termina por ?>

En cambio, en la revista empezaba correctamente por <? Y acababa por ¿> o >?

Este error no fue cometido por el autor del artículo , se produjo en el proceso de maquetado.

Con la nueva Web de la revista TODO el código original de los artículos será puesto a disposición pública para una mayor comprensión del mismo.

Pedimos disculpas a nuestros lectores.

SERVIDOR DE HXC
MODO DE EMPLEO

- Hack x Crack ha habilitado tres servidores para que puedas realizar las prácticas de hacking.

- Las IPs de los servidores de hacking las encontrarás en EL FORO de la revista (www.hackxcrack.com). Una vez en el foro entra en la zona COMUNICADOS DE HACK X CRACK (arriba del todo) y verás varios comunicados relacionados con los servidores. No ponemos las IP aquí porque es bueno acostumbrarte a entrar en el foro y leer los comunicados. Si hay alguna incidencia o cambio de IP o lo que sea, se comunicará en EL FORO.

- Actualmente tienen el BUG del Code / Decode. La forma de "explotar" este bug la explicamos extensamente en los números 2 y 3. Lo dejaremos así por un tiempo (bastante tiempo ;) Nuestra intención es ir habilitando servidores a medida que os enseñemos distintos tipos de Hack.

- En los Servidores corre el Windows 2000 con el IIS de Servidor Web. No hemos parcheado ningún bug, ni tan siquiera el RPC y por supuesto tampoco hemos instalado ningún Service Pack. Para quien piense que eso es un error (lógico si tenemos en cuenta que el RPC provoca una caída completa del sistema), solo decirte que AZIMUT ha configurado un firewall desde cero que evita el bug del RPC, (bloqueo de los puertos 135 (tcp/udp), 137 (udp), 138 (udp), 445 (tcp), 593 (tcp)). La intención de todo esto es, precisamente, que puedas practicar tanto con el CODE/DECODE como con cualquier otro "bug" que conozcas (y hay cientos!!!). Poco a poco iremos cambiando la configuración en función de la experiencia, la idea es tener los Servidores lo menos parcheados posibles pero mantenerlos operativos las 24 horas del día. Por todo ello y debido a posibles cambios de configuración, no olvides visitar el foro (Zona Comunicados) antes de "penetrar" en nuestros servidores.

- Cada Servidor tiene dos unidades (discos duros duros):
* La unidad c: --> Con 40GB y Raíz del Sistema
* La unidad d: --> Con 40GB
* La unidad e: --> CD-ROM

Nota: Raíz del Servidor, significa que el Windows Advanced Server está instalado en esa unidad (la unidad c:) y concretamente en el directorio por defecto \winnt\ Por lo tanto, la raíz del sistema está en c:\winnt\

- El IIS, Internet Information Server, es el Servidor de páginas Web y tiene su raíz en c:\inetpub (el directorio por defecto)

Nota: Para quien nunca ha tenido instalado el IIS, le será extraño tanto el nombre de esta carpeta (c:\inetpub) como su contenido. Pero bueno, un día de estos os enseñaremos a instalar vuestro propio Servidor Web (IIS) y detallaremos su funcionamiento.

De momento, lo único que hay que saber es que cuando TÚ pongas nuestra IP (la IP de uno de nuestros servidores) en tu navegador (el Internet explorer por ejemplo), lo que estás haciendo realmente es ir al directorio c:\Inetpub\wwwroot\ y leer un archivo llamado default.htm.

Nota: Como curiosidad, te diremos que APACHE es otro Servidor de páginas Web (seguro que has oído hablar de él). Si tuviésemos instalado el apache, cuando pusieses nuestra IP en TU navegador, accederías a un directorio raíz del Apache (donde se hubiese instalado) e intentarías leer una página llamada index.html ... pero... ¿qué te estoy contando?... si has seguido nuestra revista ya dominas de sobras el APACHE ;)

Explicamos esto porque la mayoría, seguro que piensa en un Servidor Web como en algo extraño que no saben ni donde está ni como se accede. Bueno, pues ya sabes dónde se encuentran la mayoría de IIS (en \Inetpub\) y cuál es la página por defecto (\Inetpub\wwwroot\default.htm). Y ahora, piensa un poco… … ¿Cuál es uno de los objetivos de un hacker que quiere decirle al mundo que ha hackeado una Web? Pues está claro, el objetivo es cambiar (o sustituir) el archivo default.html por uno propio donde diga "hola, soy DIOS y he hackeado esta Web" (eso si es un lamer ;)

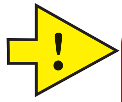
A partir de ese momento, cualquiera que acceda a ese servidor, verá el default.htm modificado para vergüenza del "site" hackeado. Esto es muy genérico pero os dará una idea de cómo funciona esto de hackear Webs ;)

- Cuando accedas a nuestro servidor mediante el CODE / DECODE BUG, crea un directorio con tu nombre (el que mas te guste, no nos des tu DNI) en la unidad d: a ser posible y a partir de ahora utiliza ese directorio para hacer tus prácticas. Ya sabes, subirnos programitas y practicar con ellos :) ... ... ¿cómo? ¿que no sabes crear directorios mediante el CODE/DECODE BUG... repasa los números 2 y tres de Hack x Crack ;p

Puedes crearte tu directorio donde quieras, no es necesario que sea en d:\mellamojuan. Tienes total libertad!!! Una idea es crearlo, por ejemplo, en d:\xxx\system32\default\10019901\mellamojuan (ya irás aprendiendo que cuanto mas oculto mejor :)

Es posiblemente la primera vez que tienes la oportunidad de investigar en un servidor como este sin cometer un delito (nosotros te dejamos y por lo tanto nadie te perseguirá). Aprovecha la oportunidad!!! e investiga mientras dure esta iniciativa (esperemos que muchos años).

- En este momento tenemos mas de 600 carpetas de peña que, como tu, está practicando. Así que haznos caso y crea tu propia carpeta donde trabajar.



MUY IMPORTANTE...

MUY IMPORTANTE!!!!! Por favor, no borres archivos del Servidor si no sabes exactamente lo que estás haciendo ni borres las carpetas de los demás usuarios. Si haces eso, lo único que consigues es que tengamos que reparar el sistema servidor y, mientras tanto, ni tu ni nadie puede disfrutar de él :(Es una tontería intentar "romper" el Servidor, lo hemos puesto para que disfrute todo el mundo sin correr riesgos, para que todo el mundo pueda crearse su carpeta y practicar nuestros ejercicios. En el Servidor no hay ni Warez, ni Programas, ni claves, ni nada de nada que "robar", es un servidor limpio para TI, por lo tanto cuídalo un poquito y montaremos muchos más :)

Lubic

crea tus propias cajas

by Aero Cool

www.aerocool.com.tw



IMPORTADOR OFICIAL
EXCLUSIVO ESPAÑA-PORTUGAL

TEL.: 902-227733 / 945-176647
www.biomag.biz / compras@biomag.biz

Los paquetes estandar son:

1. LUBIC-3519-BL	4. LUBIC-4480-BL	7. LUBIC-AIRPLANE-BL
2. LUBIC-3519-BK	5. LUBIC-4480-BK	8. LUBIC-AIRPLANE-BK
3. LUBIC-3519-SV	6. LUBIC-4480-SV	9. LUBIC-AIRPLANE-SV

NUEVA TECNOLOGIA DE TUBO SUPERCONDUCTOR

Serie Deep Impact
DP-102

Características

1. Alta conductividad térmica
2. Totalmente fabricado en cobre con un tubo superconductor para un maximo rendimiento
3. Super Diseño circular
4. DP-102 ofrece una solución de doble ventilador - pueden usarse uno o dos ventiladores.
5. Incluye dos sets de adaptadores de ventilador de 70mm a 80mm.
6. El disipador puede ser rotado 360 grados y los ventiladores pueden ser instalados en cualquier posición.
7. DP-102 puede ser "Universal" compatible con AMD e Intel P4

1 ventilador

Aplicaciones

AMD: Athlon XP 3600+ y superiores
Intel: P4 Socket 478 3.6Ghz y superiores
(P4 sólo Versión Universal)

Maximo Diseño

2 ventiladores

Disipador

Dimensiones =Tubo -100 mm, 36+ láminas - dia. 66mm
Material =Tubo Superconductor + Láminas de Cobre

¿Aburrido de las mismas cajas de ordenador de siempre?

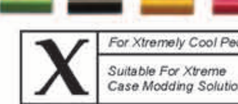
¡¡Ahora, transforma tu ordenador "cuadrado" en cualquier cosa!! Presentamos las nuevas cajas "LUBIC". Gracias al concepto modular de "LUBIC", dispones de todo lo necesario para crear tu propio diseño de caja de ordenador con los elementos de aluminio incluidos en el producto

Planifica cuidadosamente el plan de construcción y los materiales para transformar tu ordenador en algo más que una simple y aburrida caja de metal.
¡¡Todo es posible con las cajas "LUBIC" !! Puedes crear una caja "Escorpión", una caja "Elefante", una caja "bombardero B-52" o cualquier cosa que imagines. También puedes crear cosas que no sean ordenadores con los modulos "LUBIC" como portaretratos, expositores, mesas...

¿Es el cielo el limite?
Con los módulos "LUBIC", ¡¡NO EXISTEN LIMITES!!!
¡¡¡Da forma a todas sus ideas!!! ¡¡Las creaciones solo estan limitadas por tu imaginación!!!...







NUEVA TECNOLOGIA DE TUBO SUPERCONDUCTOR

Serie High Tower
HT-101

Características

1. Alta conductividad térmica
2. Totalmente fabricado en cobre con 3 tubos superconductores para un rendimiento extremo.
3. HT-101 ofrece una solución de doble ventilador - pueden usarse uno o dos ventiladores.
4. Carcasa de plástico azul sensible a la luz ultravioleta.
5. Ventilador con 4 LEDS UltraVioleta - innovadoras aspas "flower shape" para super rendimiento y silencio extremo.

Velocidad: 2500 RPM
Caudal: 29.6 CFM
Ruido: 22 DBA

Aplicaciones

AMD: Athlon XP 3600+ y superiores
Intel: P4 Socket 478 3.6Ghz y superiores

Disipador

Dimensiones =Altura -113 mm, 31 láminas, 76 x 50mm
Material =Tubo Superconductor + Láminas de Cobre

¡¡Reactivo UV!!